НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації) 121, інженерія програмного забезпечення (інженерія програмного забезпечення комп’ютерних систем) _____
(код і назва спеціальності)

_____

на тему: Автоматизована система зберігання та обробки данних з використанням алгоритмів підсумкового аналізу_____

Виконав (-ла): студент (-ка) __2__ курсу, групи ___ІТ-83мп_____
(шифр групи)

Сидько Артем Юрійович_____ _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент кафедри АУТС, к.т.н, доцент Катін П.Ю. _____
_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) Інформатики та обчилювальної техніки
(повна назва)

Кафедра Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 121, інженерія програмного забезпечення
(код і назва спеціальності)

(інженерія програмного забезпечення комп'ютерних систем)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Сидьку Артему Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Автоматизована система зберігання та обробки данних з використанням алгоритмів підсумкового аналізу

науковий керівник дисертації доцент кафедри АУТС, к.т.н, доцент Катін П.Ю.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження автоматизація процесу зберігання та обробки даних з використанням інструментів аналітики

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) система зберігання та обробки даних з налаштованими інструментами ведення аналітики

5. Перелік завдань, які потрібно розробити ознайомитися з архітектурними моделями сучасних систем обробки та зберігання інформації, сформувати вимоги до системи та описати сценарії використання, реалізувати алгоритми зберігання та обробки даних, інтегруватись з системою розпізнавання образів, використовуючи отримані дані розробити власну системи зберігання та обробки даних з використанням алгоритмів підсумкового аналізу

6. Орієнтовний перелік ілюстративного (графічного) матеріалу діаграма послідовності процесу інтеграції, структурна діаграма процесу, інтеграції, ER-діаграма

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання 5 вересня 2019 року

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка

Студент _____ Сидько А.Ю.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Катін П.Ю.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Магістерська дисертація складається з 105 сторінок, 19 зображень, 31 таблиця, 16 посилань на використані джерела та 8 додатків.

Актуальність теми обраної для написання дисертації полягає в автоматизації процесів управління системою зберігання та обробки даних є дуже важливим питанням, що постає рано чи пізно перед кожним користувачем подібної системи. На даний момент існує багато аналогів, проте більшість із них мають суттєві недоліки, а також зазвичай не мають ефективної аналітик або ж інтеграцій з системами розпізнавання.

Метою дисертації є розробка автоматизованої системи зберігання та обробки даних з алгоритмами підсумкового аналізу, яка дозволить легко підвищити ефективність корпоративних бізнес процесів.

Об’єктом дослідження дисертації є автоматизація алгоритмів обробки даних з подальшим їх використанням.

Предметом дослідження є моделі зберігання та обробки даних з використанням хмарних технологій.

Під час розв’язку поставлених задач було використано методи наукового пізнання, а саме: порівняння, вимірювання, індукція та дедукція. За допомогою цих методів було досягнено значних результатів у оптимізації існуючих процесів, шляхом попереднього аналізу існуючих рішень та аналогів, що дало можливість визначити основні підходи до ефективної автоматизації процесів.

Результати роботи можуть бути використані у системах взаємодії даних в якості окремих модулів, або ж як самостійна система обробки та зберігання даних

Ключові слова: база даних, аналітика, сервіс, хмарне середовище

SUMMARY

The master's thesis consists of 105 pages, 19 images, 31 tables, 16 references to used sources and 8 applications.

The relevance of the topic chosen for the dissertation writing is to automate the processes of managing the storage and data processing system is a very important issue that arises sooner or later before every user of such a system. There are many analogues at the moment, but most of them have significant drawbacks and usually do not have an effective analyst or integration with recognition systems.

The purpose of the dissertation is to develop an automated system of storage and processing of data with algorithms of the final analysis which will allow to increase easily efficiency of corporate business processes.

The object of the dissertation research is automation of data processing algorithms with their further use.

The subject of the study is data storage and processing models using cloud technologies.

The methods of scientific cognition, namely, comparison, measurement, induction and deduction, were used in solving the tasks. With these methods, significant results have been achieved in the optimization of existing processes, through preliminary analysis of existing solutions and analogues, which made it possible to identify the main approaches to the effective automation of processes.

The results of the work can be used in data systems as separate modules, or as a stand-alone data processing and storage system

Keywords: database, analytics, service, cloud environment

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface – інтерфейс програмної взаємодії.

SDK – software development kit – набір інструментів для розробки.

Queue – черга повідомлень в середовищі Azure Service Bus.

БД – база даних.

Instance – екземпляр налаштованої системи.

GUID – глобальний унікальний ідентифікатор в рамках системи.

HTTP – hypertext transfer protocol – протокол передачі гіпертексту.

HTTPS – hypertext transfer protocol secured – наступна версія HTTP протоколу з більш захищеним протоколом передачі.

ВСТУП

Світ невпинно женеться за автоматизацією абсолютно усього, що існує в нашому сьогоденні. Але якщо поглянути у корінь даної тенденції, то ми зрозуміємо, що насправді світ женеться за збільшенням прибутків та зменшенням витрат. Частково на такі потреби відповідає автоматизація, проте не завжди це так. Починаючи впровадження автоматизації, керівники та менеджери намагаються вирахувати ефективність даного впровадження і робиться це різноманітними алгоритмами, в тому числі алгоритмами підсумкового аналізу, що простіше називають аналітикою. Наймають чималу кількість різних спеціалістів лише для того, щоб вирахувати ефективність та оцінити, на скільки насправді покращилось становище.

Неможливо аналізувати дані без самих даних, що мають тенденцію накопичуватись. Великі компанії та організації, що існують хоча б декілька років мають величезні об’єми інформації, яку вони мають необхідність оброблювати. Тому постає необхідність у ефективній системі обробки та зберігання даних, яка була б зручною, збільшувала прибутки та надавала корисну аналітику щодо процесу роботи компанії.

Також однією з тенденцій сьогодення є впровадження систем з використанням штучного інтелекту, але часто інформація, яку він надає, залишається не використаною через відсутність алгоритмів взаємодії з даними системами. Системи існують, частково використовуються, але більша частина того, що вони роблять залишається не використаною

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Зберігання даних, як ми вже зазначили, надзвичайно важливе завдання сьогодення. На вирішення даного питання було витрачено чимало зусиль різних розробників, архітекторів та аналітиків з усього світу. Є чимало вдалих і не дуже варіантів реалізацій схожих за функціоналом систем. Перед розглядом конкретних рішень, варто зазначити основні вимоги до подібних систем.

Системи зберігання та обробки даних мають бути зручними, інтуітивно зрозумілими для кінцевих користувачів, швидко оброблювати великі обсяги інформації, мають бути масштабованими, а також мати необхідний додатковий функціонал, що визначений конкретною сферою використання. Перші два пункти стосуються користувацького досвіду та залучення до користування системою великої кількості користувачів. Велика кількість нових користувачів мають мати можливість швидкого навчання, адже подібні системи мають забезпечувати одночасний доступ для багатьох користувачів. Велика кількість користувачів зазвичай передбачає додатковий час на навчання користуванням системою. В умовах високонавантажених систем є ефективним використання інтуітивно зрозумілого дизайну, щоб зменшити витрати часу для нових користувачів на пристосування до системи. Не менш важливим є зручність використання, адже тоді підвищується і ефективність використання. Все, що можна автоматизувати варто автоматизувати.

Системи зберігання та обробки інформації зазвичай пов’язані не лише із зберіганням інформації, але й і з обробкою та подальшим використанням саме обробленої інформації або ж результатів обробки, що формують нові інформаційні ресурси. Оскільки методи обробки інформації змінюються з часом, а також враховуючи можливі технологічні зміни конкретного користувача, подібні системи мають бути гнучкими для додання додаткових алгоритмів обробки, зберігання, маніпулювання данними тощо.

Щодо додаткового функціоналу варто зазначити, що його можна поділити на два типи. Загальновживаний, що є спільним для більшості користувачів та специфічний, що

визначений особливими умовами або цілями використання системи. До загальновживаних можна віднести ролі безпеки, їх налаштування, доступ до аналітики щодо використання доступних ресурсів тощо. До специфічних можна віднести особливі алгоритми обробки та подальшої передачі даних, а також відображення кінцевих або ж проміжних результатів. Додатковий функціонал зазвичай формується на початку проектування системи, але даний пункт найчастіше модифікується після впровадження систем у експлуатацію.

Виходячи з вищеперерахованого, можна зробити висновок, що наша система за своїми вимогами схожа на стандартну CRM-систему. Єдиного визначення такої системи не існує, але можна зрозуміти її сутність переклавши назву. Управління відносинами з клієнтами (з англійської Customer Relationship Management) — поняття, що охоплює концепції, котрі використовуються компаніями для управління взаємовідносинами зі споживачами, включаючи збір, зберігання й аналіз інформації про споживачів, постачальників, партнерів та інформації про взаємовідносини з ними. Зазвичай під CRM системою мають на увазі систему, що відповідає певним вимогам або ж має певні принципи. До основних принципів таких систем можна віднести наступне:

- наявність єдиного сховища інформації, звідки в будь-який момент доступні усі відомості про усі випадки взаємодії з клієнтом;
- синхронізація управління множинними каналами взаємодії;
- постійний аналіз зібраної інформації про клієнтів та прийняття відповідних організаційних рішень — наприклад, «сортування» клієнтів на основі їхньої значимості для компанії.

Дані принципи спрямовані на підвищення ефективності роботи з клієнтами, проте в їх основі лежить саме ефективність обробки даних у великих кількостях. Саме тому в якості основних аналогів ми вирішили розглянути саме CRM системи.

1.1 Система amoCRM

В якості першого існуючого рішення ми розглянемо CRM систему amoCRM. Даний продукт є популярним рішенням у нашій країні та країнах СНГ завдяки багатьом своїм корисним функціям, що розглянемо детальніше нижче. Варто зазначити, що даний продукт на ринку вже 10 років і має багато технологічних рішень, що були створені та доповнені в реальних умовах роботи системи. Чимала кількість бізнес-користувачів та клієнтів, що в процесі роботи давали зворотній зв’язок та корисні ідеї для впровадження або ж змінення існуючих алгоритмів, допомогли розвинути систему до достатньо високого рівня. На рахунку даного продукту більше 50 000 впроваджень не лише українського ринку. Перейдемо до плюсів даного рішення:

- amoCRM має зручний інтерфейс, що зрозумілий для користувача, інтуїтивно зрозумілий дизайн дозволяє значно пришвидшити навчання нових користувачів.
- даний продукт має реалізовані бізнес-процеси першої необхідності для будь-якого клієнта «з коробки». Серед таких процесів можна виокремити розсилки електронних листів, можливість фіксування телефонних дзвінків тощо;
- існує можливість багатокористувацького одночасного доступу для роботи з системою;
- реалізує реальні вимоги клієнтів та має підтримку у вирішенні термінових питань цілодобово;
- система постійно оновлюється, а команда розробників публікує глобальні оновлення декілька разів на рік;
- резервне копіювання даних системи;
- зручна система пошуку та організації даних.

Окремо варто зупинитись саме на системі зберігання даних у цій системі. У ній є попередньо створені сутності та відповідно база даних. Дані сутності мають основні атрибути, які найчастіше використовуються у них, а також зв’язки між логічно пов’язаними сутностями. Подібне рішення не є новим та допомагає зекономити чимало

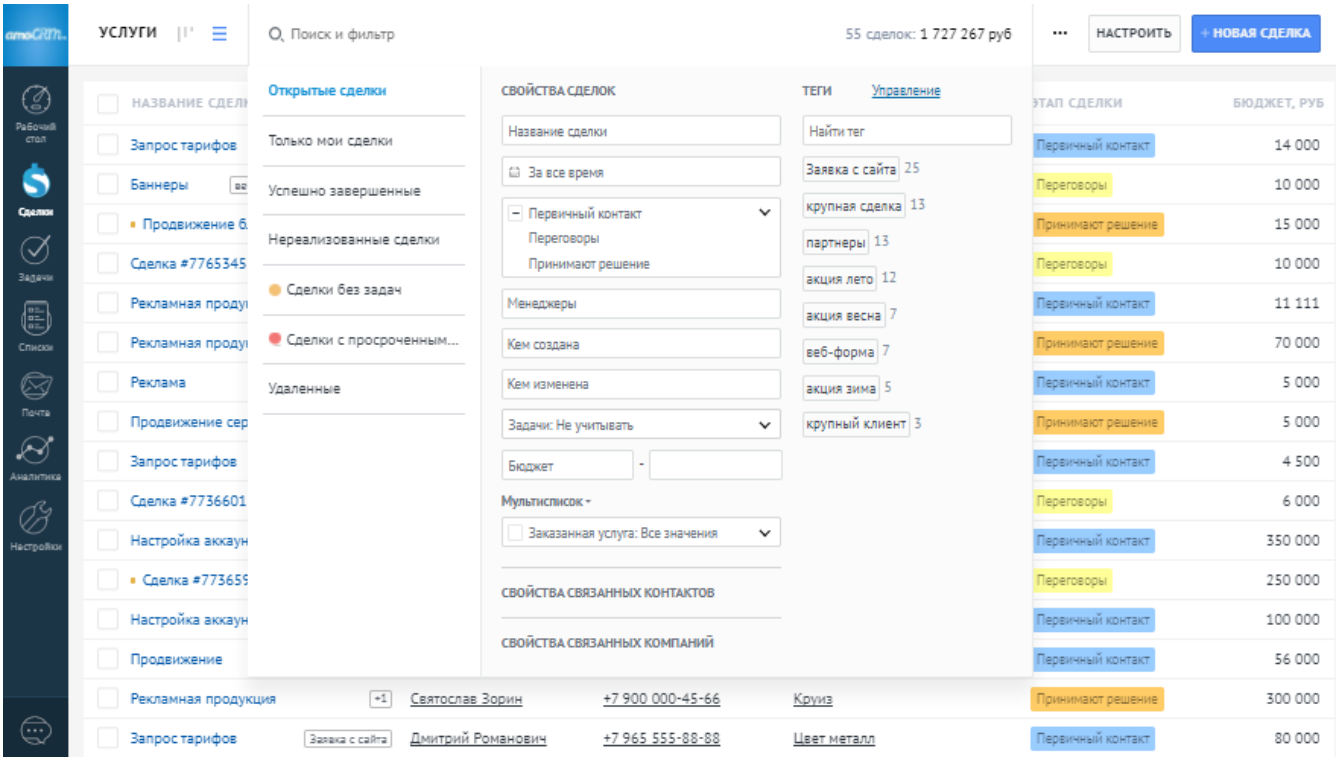

Рисунок 1.1 – інтерфейс користувача amoCRM

часу при розробці, адже переважна більшість компаній та користувачів подібних систем працюють з однаковими за змістом сутностями. Яскравим прикладом є сутність «Контакт», яка використовується у переважній більшості систем. Дана сутність описує фізичну або юридичну особу, з якою користувач співпрацює. У даної сутності є стандартні атрибути, такі як: назва або ім’я, вік, час співпраці, канал комунікації, номер телефону, електронна адреса тощо. Подібних сутностей чимало, їхнє використання не обов’язкове, але часто є доцільним.

1.2 Система bpm`online

Даний продукт є українським, створений та підтримуваний компанією TerraSoft. Компанія позиціонує дане рішення, як рішення для бізнесу, що має широкий функціонал для кінцевого користувача. Кінцевим користувачем в даному контексті є працівники компанії, в якій впроваджена система, а саме менеджери та працівники відділу продажу.

Загалом система є популярною в межах країни, проте також має впровадження у закордонних партнерів. Таким чином число впроваджень досягає 3 тисяч в різних регіонах країни та поза її межами.

В більшості користувачами даної системи є представники середнього та великого бізнесу, при чому других переважна більшість. Це пояснюється тим, що система не є дешевою та має багато реалізованих бізнес-процесів, що завжди наявні у великих компаніях і рідше у середніх та малих. Бізнес процеси самі по собі можна умовно поділити на дві категорії: загальновживані та вузького застосування. Більша частина процесів, що реалізовані в даній системі, є загальновживаною, але з великою кількість умов та додаткових прошарків, що рідко використовуються представниками малого та середнього бізнесу.

Дана система попри свою спрямованість має багатий перелік як переваг так і недоліків, спершу зазначимо позитивні риси системи:

- Широкий функціонал. Система дійсно має багато реалізованих сценаріїв взаємодії.
- Приємний інтерфейс, простий, зручний, з гнучким налаштуванням.
- Наявність своєї мови, гнучкі налаштування різних етапів роботи і звітів.
- Наявність BPM системи. Навіть в тому вигляді, яка є, BPM-система є плюсом, так як дозволяє працювати з продажами як з бізнес-процесами

Основні мінуси:

- Складність налаштування і доопрацювання системи.
- Складна і заплутана документація.
- Складна API з великим числом обмежень.
- Складна інтеграція як з даною системою зовнішніх систем, так і навпаки.
- Якість роботи підтримки на незадовільному рівні.

Крім того, до слабких сторін системи bpm'online можна віднести велику кількість зайвих можливостей. Розробники не тільки «нав'язують» певний бізнес-процес для роботи з можливостями і клієнтами, але також практично на кожному етапі роботи пропонують якісь додаткові функції, які не стільки допомагають, скільки відволікають від роботи або вносять плутанину.

Резюмуючи огляд даного продукту можна зробити висновок, що система багата як недоліками, так і перевагами, але для нашої системи на даному етапі важливо запозичити плюси й виокремити вимоги до корисного та зручного програмного забезпечення. На даному прикладі варто зазначити, що багатофункціональність може бути надмірною та варто краще продумати можливості та функції системи, щоб не нагромаджувати систему зайвими елементами, які не будуть використовуватись. Подібні елементи краще поєднати з іншими процесами або зробити додатковий функціонал, що не буде заважати роботі з основним.

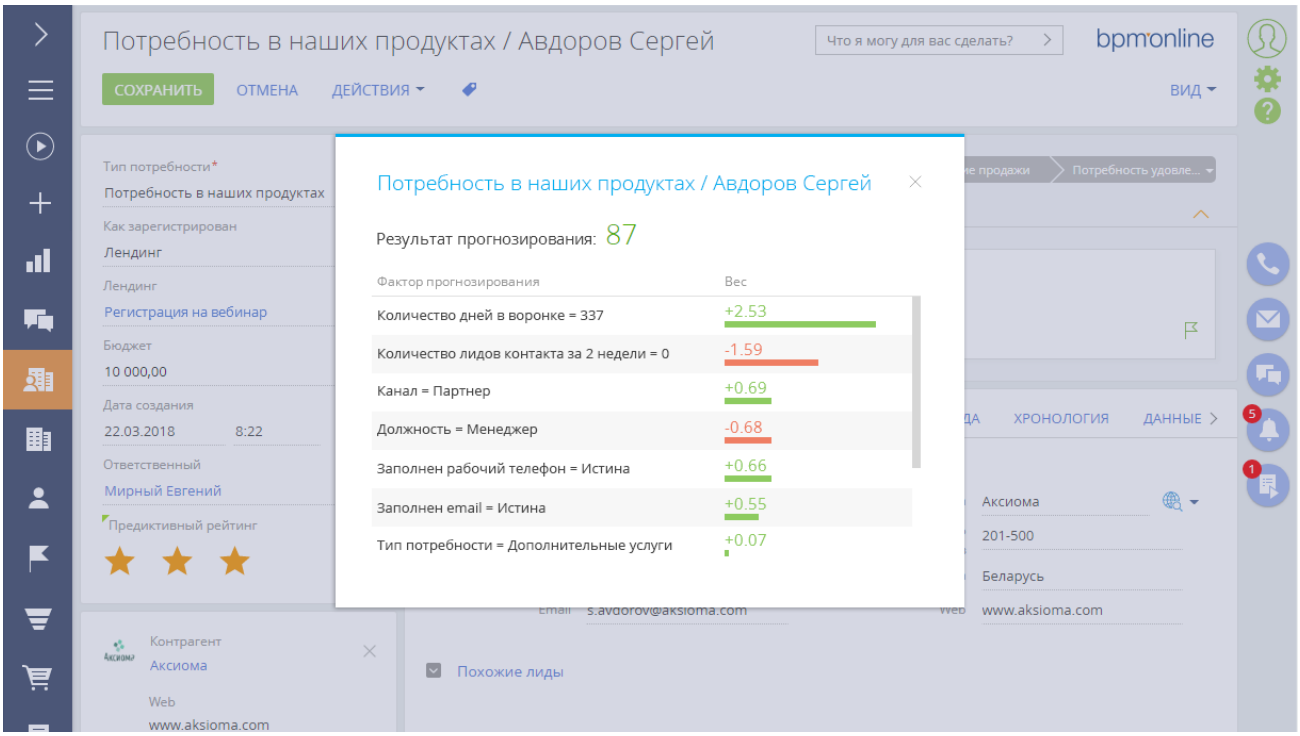

Рисунок 1.2 – інтерфейс користувача bpm`online

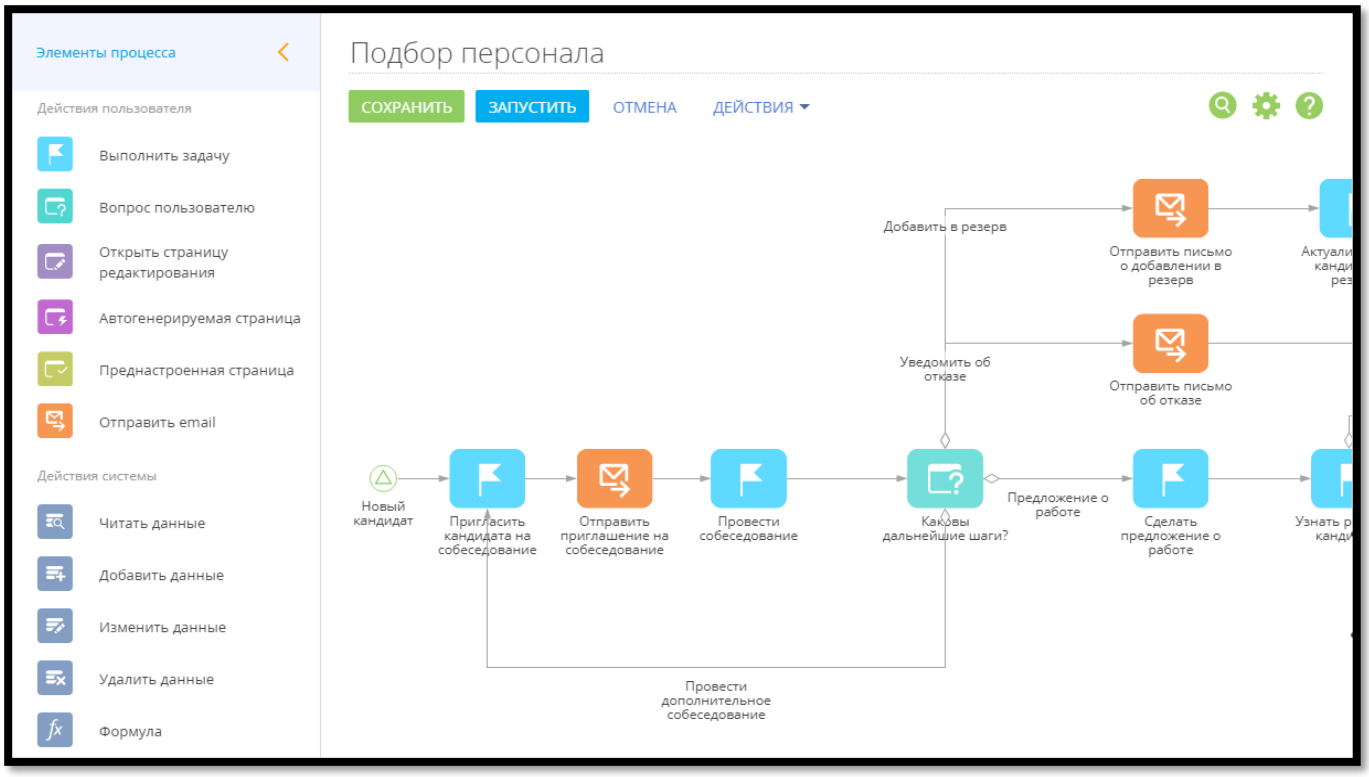

Рисунок 1.3 – налаштування бізнес процесів у системі bpm`online

Незважаючи на чисельні технологічні недоліки системи її обирають також за її простоту та зрозумілість інтерфейсу, який можна налаштовувати під окремого користувача.

Також не можна опустити той факт, що дана система написана з використанням мови, що була розроблена компанією, що дає певні переваги у розробці бізнес процесів, проте в глобальному плані збільшують терміни розробки, а також нова мова потребує часу, для вивчення для спеціалістів не вузького профілю для подальшої розробки, налаштування або ж підтримки системи.

1.3 Salesforce Sales Cloud

Salesforce Sales Cloud - популярний продукт CRM із продуктів Salesforce, який включає окремі рішення для хмарного середовища, сервісу, маркетингу та аналітики. Система допомагає відстежувати всю взаємодію із клієнтами та інформацію в одному

місці та надавати додаткові результати. Існує можливість використовувати такі функції, як управління партнерами, дані про продажі, управління потенційними клієнтами та автоматизація маркетингу, щоб дотримуватися цих потенційних результатів і виховувати їх для конверсії.

Постачальник пропонує чудовий безкоштовний пробний план, який дозволяє спершу спробувати всі основні функції.

Команда з продажу може продуктивно використовувати додаток, будь то на місцях чи в офісі. Постачальник пропонує мобільні програми для пристроїв iOS та Android із спеціальними звітами та даними в режимі реального часу з інформаційних панелей. Відмінна особливість - Feed First, яка дозволяє швидко переглядати найважливіші дані на основі налаштувань. Ви можете розширити функціональність, додавши більше інструментів з AppExchange. Система допомагає отримувати глибокі звіти з даних клієнтів у режимі реального часу, створювати важливі моделі території та будувати точні прогнози продажів.

CRM Salesforce створений для збільшення рахунків, закриття угод та швидшого пошуку клієнтів. Це загальні переваги, якими користуються компанії під час використання CRM Salesforce для управління відносинами з клієнтами:

Ви можете отримати повне уявлення про клієнта, наприклад, історію діяльності, спілкування, дискусії та соціальні згадки за допомогою управління контактами. Відстежувати угоди та шукати конкурентну інформацію за допомогою модуля «Співпраця з продажу». Точно так само наявне управління ефективністю продажів, що дозволяє встановлювати цілі на основі показників та посилювати виграшні показники продажів, щоб мотивувати здорову конкуренцію в команді.

Плюси даної системи:

- існує величезний спектр віджетів звітів, які дозволяють аналізувати результати діяльності компанії з різних точок зору;

- хмарне рішення, компаніям потрібно менше ресурсів в офісі для ведення бізнесу. Кожен працівник може отримати доступ до CRM з усього світу та зробити свій внесок у загальну продуктивність;
- сумісність, Salesforce підтримують CRM у всіх основних браузерах. Це робить дуже простий у використанні функцію соціальних медіа, як Chatter, яка може допомогти бізнесу знайти співпрацю. Схожість на популярні соціальні медіа робить його дуже знайомим та легким для розуміння користувачам. Такі функції, як перегляд новин, оновлення статусу, обмін посиланнями та завантаження файлів, доступні прямо на інформаційній панелі;
- швидкий старт, швидке розгортання, що не потребує додаткового програмного забезпечення;

До мінусів в свою чергу можна віднести наступні фактори:

- дороге налаштування. Якщо є необхідність використання сторонніх рішень або налаштування, потрібно мати на увазі додаткові витрати;
- складність, маленьким компаніям, які не мають спеціалізованої команди з продажу або маркетингу, може не знадобитися таке складне і потужне рішення, як Saleforce;
- інтерфейсні перетворення, кожне оновлення або випуск може різко змінити порядок розміщення на інформаційній панелі або вкладках. Таким чином, процес адміністрування може сильно ускладнитись;
- складна технічна підтримка, дуже важко звернутися до технічної служби Salesforce у разі виникнення питань. Іноді для зв’язку з відповідальною особою може знадобитися кілька днів. Але це не означає, що ваша проблема буде вирішена. Таким чином, багато організацій вважають за краще працювати з сторонніми компаніями;
- складне звітування. Щоб отримати важливу інформацію про, ви

повинні бути готові зробити декілька кроків із доступних для отримання необхідних даних.

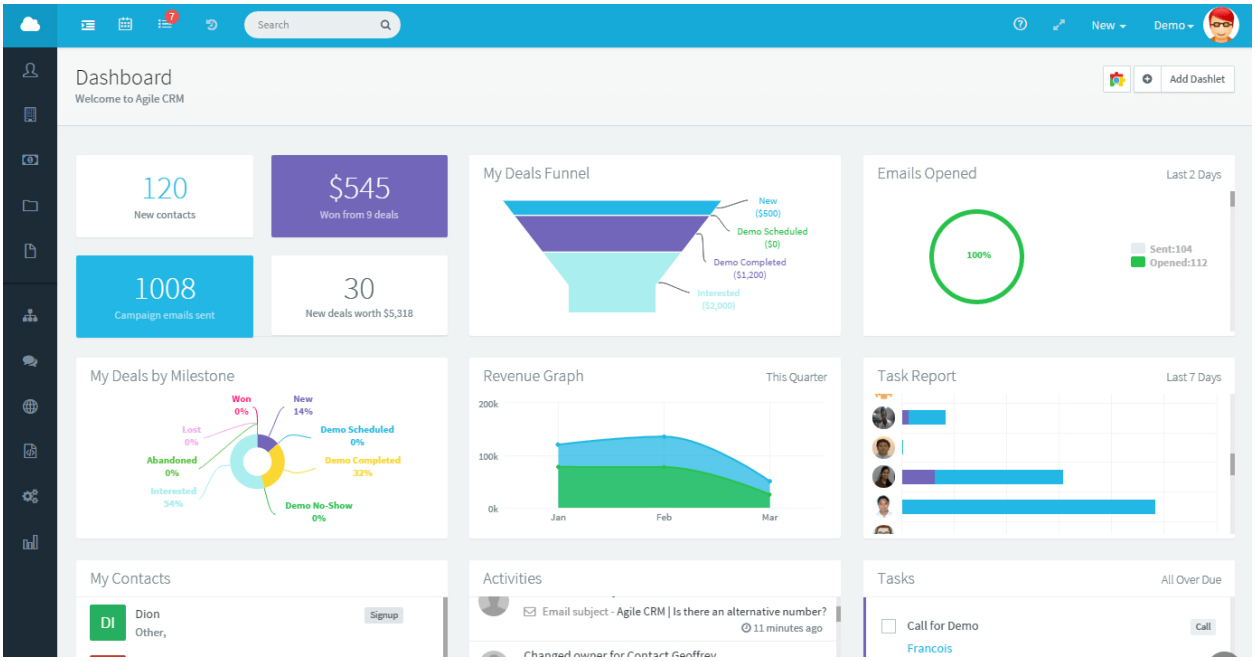

Рисунок 1.4 – інтерфейс користувача в системі salesforce crm

Резюмуючи огляд даної системи, можна зазначити, що дана система з усіх розглянутих найбільш велика та насичена функціональними особливостями. Програмний продукт є іноземним, використовує напрацювання іншої відомої компанії Oracle. Враховуючи все вищеперераховане можна зазначити, що використання вже існуючих алгоритмів та засобів, що довели свою ефективність з часом, є доцільним у системі з багатим функціоналом. Також варто звернути увагу на систему аналітики та звітування, налаштування її має бути простим та зрозумілим для користувача, а головне не має витрачати багато часу та кроків для досягнення кінцевого результату. Також варто розглянути можливість автоматизації аналітики. Також вкотре варто відмітити важливість зручного та незмінного в основі інтерфейсу, щоб не збивати користувачів та не змушувати перенавчатись. Загалом система має чимало переваг, які варто врахувати, найголовніші з них це варіативність, простота роботи, легкість налаштування реалізованого функціоналу.

Висновки до розділу

Провівши огляд існуючих аналогів нашої системи, нами були виокремлені сильні сторони та проблемні області, на які важливо звернути увагу при подальшій розробці програмного продукту. Щодо кожного аналогу був проведений детальний аналіз особливостей та досліджено відгуки реальних кінцевих користувачів даних систем.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Аналіз вимог (також називається розробка вимог) - це процес визначення очікувань користувачів щодо нових продуктів. Ці потреби (називаються вимогами) необхідно кількісно визначити та деталізувати. У розробці програмного забезпечення ці вимоги часто називають функціональними специфікаціями. Аналіз вимог є важливим аспектом управління проектами.

Під час цієї фази розробки програмного забезпечення корисним є спілкування із користувачами системи, щоб визначити конкретні очікування щодо функціональності системи, вирішити будь-які конфлікти чи неоднозначності, необхідні для різних користувачів або груп користувачів, та всі аспекти процесу розробки проекту.
Документ від початку до кінця. Основна увага зосереджується не на спробі встановити очікування клієнтів, які відповідають вимогам, а на забезпеченні відповідності кінцевої системи чи продукту потребам клієнта.
Вимоги поділяються на функціональні вимоги та нефункціональні вимоги. Аналіз вимог можна розділити на три етапи.

- Збирайте вимоги, включаючи аналіз збору додатків, огляд існуючих рішень та спілкування з клієнтами та майбутніми користувачами.

- Аналіз вимог. На цьому етапі визначте основні характеристики та характеристики системи. Послідовність, конфлікт між вимогами, співвідношення між вимогами та їх послідовність.

- Документація на вимоги. Документація щодо вимог повинна бути офіційно записана у міру необхідності.

Всі вимоги поділяються на функціональні та нефункціональні.

Функціональні вимоги

Функціональні вимоги - це вимоги, що підкреслюють функціональність, необхідну для програмного продукту. Загалом, функціональність залежить від поведінки системи та вхідних та вихідних параметрів. Взаємодія даних, бізнес-процес, взаємодія з користувачем або інша специфічна системна функція.

Основні функціональні вимоги:

- Система повинна приймати на вхід дані від зовнішніх систем та має мати можливість оброблення даних швидко та ефективно.
- Системі потрібно зберігати дані у спеціально налаштованому сховищі даних з чітко продуманою структурою.
- Система повинна обробляти напівструктуровані метадані та зберігати їх у форматі, який підтримує основні маніпуляції з даними (фільтрування, сортування, групування, агрегація).
- У системі мають бути наявні механізми проведення ефективної аналітики.
- Система повинна бути здатна інтегруватися із зовнішніми службами, використовуючи новітні протоколи передачі даних.
- Система повинна бути доступною в Інтернеті.
- Плюсом системи може бути розміщення її компонентів у хмарному середовищі.
- Система має бути модульною для забезпечення зручного її розгортання та легкої інтеграції з іншими системами.

Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики програмної системи. Це набір критеріїв, що використовуються для оцінки функціональності певної системи. Нефункціональні вимоги важливі для забезпечення доступності та ефективності всієї програмної системи. Недотримання нефункціональних вимог може призвести до того, що система не задовольняє потреби користувача. Опис нефункціональних вимог так само важливий, як і опис функціональних вимог.

Основні нефункціональні вимоги:

- Система має швидко формувати звіти.
- Кількість даних від зовнішніх систем має не перевищувати 300000 на день.
- Інтеграційні точки запобігають несанкціонованому втручанню.
- Контакт інтеграції вимагає щонайменше 99,9% SLA (наявність).
- Система повинна бути економічно доцільною порівняно з конкурентами.

За замовчуванням для імпорту даних через API HTTP потрібна інтеграція.

- Мають бути оброблені всі можливі варіанти помилок, система має бути покрита тестами на стільки, на скільки це є можливими.

Висновок до розділу

У цьому розділі описані загальні функціональні та нефункціональні вимоги системи. Системні вимоги - важлива особливість, яка визначає якість системи та відповідність вимогам користувача. Для всіх типів вимог ігнорування вимог загрожує цілісності рішення. Функціональні та нефункціональні вимоги тісно пов'язані з багатьма відносинами.

Кожне рішення підвищує ефективність завдяки вичерпному переліку вимог, зібраних під час запуску та впровадження. Вимоги можна розділити на дві категорії: основні потреби та основні потреби. Основні вимоги випливають із аналогії з "функціональними вимогами", безпосередньо пов'язаними з рішенням. Але так звані основні.

СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Ролі користувачів у системі

Користувачі системи діляться на 3 типи: користувач що працює з даними(оператор); користувач, що імпортує дані в систему (імпортер); користувач-адміністратор системи. В даному випадку користувач, що імпортує дані в систему це буде система, що налаштована один раз і в подальшому не буде видозмінюватись.

Функції користувачів:

1) Імпортер — завантаження даних у систему (передача зображення та метаданих);
2) Оператор — аналіз та обробка даних та звітів із системи;
3) Адміністратор — отримання внутрішньої статистики роботи системи, налаштування доступів до системи, моніторинг коректної роботи системи.

Опис сценаріїв використання

Для опису дій використовуються наступні позначення: для дій користувача – літера К та номер дії; для дій системи – літера С та номер дії.

Детальний опис сценаріїв використання представлений на таблицях 3.1 – 3.6.

Таблиця 3.1 – Опис сценарію використання «Імпорт даних у систему»

Назва сценарію	Імпорт даних у систему
Спричиняє події	Зчитування події, що запускає процес імпорту
Короткий опис	Для запуску процесу імпорту даних потрібно змінити дані певного поля спеціальної сутності або ж натиснути на кнопку синхронізації в інтерфейсі користувача
Частота	Можливе постійне використання

Назва сценарію	Імпорт даних у систему
Актори	Імпортер

Продовження таблиці 3.1

Передумови	Мають бути виконані умови такі як зареєстрованість джерела отримання у системі, певний бізнес етап у статусі джерела
Вихідні умови	Дані у сховищі, оброблені та доступні для формування звіту
Опис дій	К1 – користувач зчитує вхідне повідомлення у правильному форматі та передає його до системи. Повідомлення має містити один/кілька об’єктів, що описують зображення з відповідними напівструктурованими метаданими. С1 – система приймає вхідне повідомлення, валідує його, зберігає дані в сховище даних, створює запис у табличному сховищі, що містить посилання на об’єкт, сервісні дані та вхідні метадані.
Очікувані умови	Створення запису про об’єкт, збереження даних до сховища

Таблиця 3.2 – Опис сценарію використання «автоматичне формування звіту системою»

Назва сценарію	Аналіз та обробка даних та звітів із системи
Спричиняє подію	Запуск процесу формування звіту
Короткий опис	Запуск системою процесу формування та збереження звіту
Актори	Система

Частота	Задається об’ємом даних, які потрібно оброблювати

Продовження таблиці 3.2

Передумови	Дані про об’єкти присутні у системі
Вихідні умови	Звіт у попередньо заданому форматі зберігається у даному форматі
Опис дій	С1 – система періодично за розкладом спричинює запуск процесу формування звіту. С2 – система перевіряє наявність даних про об’єкти у систем. С3 – система вмикає та підготовлює механізм формування звіту. С5 – отриманий результат зберігається у вигляді файлу з розширенням, зазначеним користувачем. С6 – після заверення обробки звіту, система вимикає механізм формування.
Очікувані умови	Збереження звіту у файлі для подальшого використання

Таблиця 3.3 – Опис сценарію використання «Формування звіту системою за вимогою»

Назва сценарію	Аналіз та обробка даних та звітів із системи
Спричиняє подію	Запуск процесу формування звіту
Короткий опис	Запуск системою процесу формування та збереження звіту
Актори	Система
Частота	Задається об’ємом даних, які потрібно оброблювати
Передумови	Дані про об’єкти присутні у системі

Продовження таблиці 3.3

Вихідні умови	Звіт у попередньо заданому форматі зберігається у даному форматі або відправляється на електронну пошту оператора, що ініціював процес
Опис дій	С1 – система періодично за розкладом спричинює запуск процесу формування звіту. С2 – система перевіряє наявність даних про об’єкти у систем. С3 – система вмикає та підготовлює механізм формування звіту. С5 – отриманий результат зберігається у вигляді файлу з розширенням, зазначеним користувачем або у вигляді об’єкту для перегляду. С6 – після заверення обробки звіту, система вимикає механізм формування.

Таблиця 3.4 – Опис сценарію використання «Отримання внутрішньої статистики роботи системи»

Назва сценарію	Отримання внутрішньої статистики роботи системи
Короткий опис	На цьому етапі адміністратор може переглянути кількість даних оброблених за певний проміжок часу та статистику по цілісності даних
Актори	Адміністратор, систем
Передумови	Дані існують у системі
Вихідні умови	Дані експортовані у необхідний формат та доступні для роботи в інших системах/для інших процесів.

Продовження таблиці 3.4

Опис дій	К1 – запускає механізм формування звіту С1 – система перевіряє наявність даних про об’єкти у систем. С2 – система вмикає та підготовлює механізм формування звіту. С3 – отриманий результат зберігається у вигляді файлу з розширенням, зазначеним користувачем або у вигляді К2 – користувач вимикає механізм формування звіту
Очікувані умови	Дані експортовані у необхідний формат та доступні для роботи в інших системах/для інших процесів.

Таблиця 3.5 – Опис сценарію використання «Моніторинг коректної роботи системи»

Назва сценарію	Моніторинг коректної роботи системи.
Короткий опис	Відбувається постійна передача результатів роботи системи у зовнішнє середовище за допомогою інтеграційних модулей, що працють завдяки передбаченим точкам інтеграції.
Актори	Система
Спричинюється подіями	Поява нових результатів передачі даних із зовнішньої системи
Передумови	Дані існують у системі; є під’єднаний інтеграційний модуль імпорту до відповідної точки інтеграції

Назва сценарію	Моніторинг коректної роботи системи.
Вихідні умови	Дані експортовані у необхідний формат та доступні для роботи в інших системах/для інших процесів

Продовження таблиці 3.5

Опис дій	С1 – система перехоплює інформацію про надходження даних від зовнішньої системи. С2 – окрім стандартного процесу збереження результатів у сховище, система надсилає дані в інші системи за допомогою модулю інтеграції.
Очікувані умови	Дані експортовані у необхідний формат та доступні для роботи в інших системах/для інших процесів

Таблиця 3.6 – Опис сценарію використання «Отримання аналітики по використанню ресурсів та об’ємах даних»

Назва сценарію	Отримання аналітики по використанню ресурсів та об’ємах даних
Короткий опис	В цьому сценарії використання користувач має змогу отримати аналітику по використанню системи.
Актори	Адміністратор
Передумови	-
Вихідні умови	Користувач отримує системний звіт про використання ресурсів за певний часовий проміжок
Опис дій	К1 – користувач формує запит по типам та критерія, що доступні у системі.

Назва сценарію	Отримання аналітики по використанню ресурсів та об’ємах даних
	С2 – окрім стандартного процесу збереження результатів у сховище, результати зберігаються у попередньо заданому форматі.

Висновки до розділу

Описані сценарії взаємодії повністю задовольняють вимогам до програмної системи. На основі даних сценаріїв можна розробити тести, що будуть засвідчувати коректну роботи системи. На основі даних таблиць та сценаріїв використання була побудована діаграма прецедентів.

3 СТРУКТУРНА СХЕМА СИСТЕМИ

Структурна схема призначена для відображення компонентів та взаємодії різних частин системи. Зазвичай така схема демонструє наявність підсистем у складі системи, а також інших компонентів, що забезпечують управління та взаємодію між чисельними багаторівневими програмними модулями.

Одним з найважливіших етапів проектування програмного продукту є розробка блок-схеми. Вона включає в себе сукупність компонентів, з яких складається система, взаємозв’язків між цими компонентами, додаткових об’єктів. Дана блок-схема дозволяє мати чітке уявлення про структуру системи, що відкриває можливості для модифікації такої у майбутньому та масштабуванні програмного продукту.

Дана система спроектована з використанням принципів концепції багатошарової архітектури. Загалом, успішно застосовуються на практиці багато різних типів архітектури. Найчастіше використовується традиційна трирівнева система, яка розділяє додаток на три рівні.

Тут слід зазначити, що багаторівнева архітектура зазвичай представляє два недосконало пов'язаних поняття, n-шар та n-ярус. Шари та яруси часто називають терміном «шар», а термін «шар» іноді використовується разом із «ярусом

Шар являє собою фізичний шар. Іншими словами, якщо говорити про трирівневу архітектуру, n-ярусна програма може бути розбита на сервер баз даних, веб-додаток на

веб-сервері та рівень браузера користувача. Це означає, що кожен рівень являє собою окремий спеціальний фізичний процес, навіть якщо сервер бази даних, веб-сервер та браузер користувача знаходяться на одному комп'ютері. Якщо мобільний додаток використовується замість десктопного клієнта, це окремий фізичний рівень.

Шари представляють собою логічні рівні. Іншими словами, є рівень доступу до даних, рівень логіки бізнес процесів, рівень представлення, рівень обслуговування тощо. У цьому випадку логічний рівень не такий, як фізичний. Тому, як правило, рівень презентації включає контролер для обробки вводу даних і подання, що відображається у веб-браузері. Тобто вона поділяється на два фізичні рівні. Загальна структурна схема трирівенвої архітектури зображена на рис. 4.1

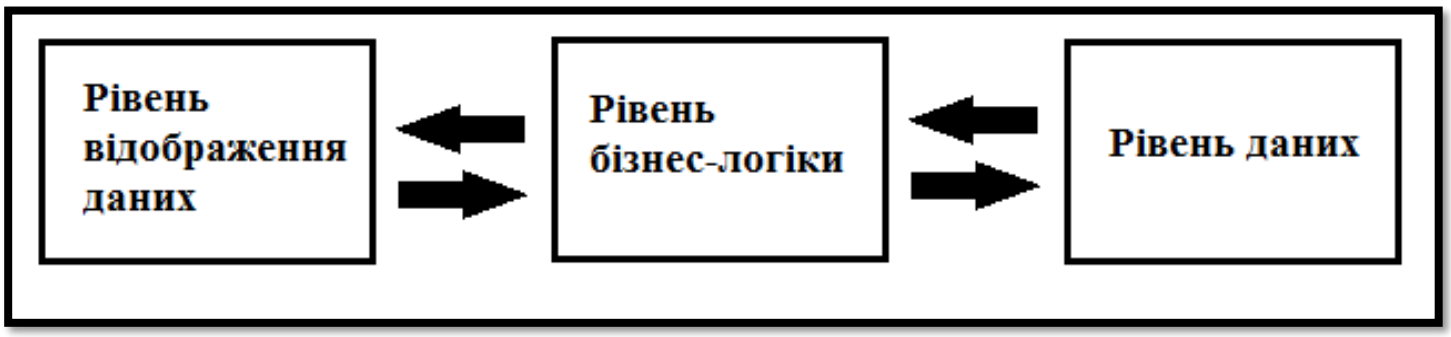

Рисунок 4.1 — Схема трирівневої архітектури

Презентаційний шар – це рівень прямої взаємодії з користувачем. Цей рівень включає компоненти інтерфейсу користувача та механізм прийому вводу від користувача. Для mpc asp.net цей рівень відображає представлення даних, усі компоненти, що входять до інтерфейсу користувача (стиль, статичний HTML, сторінка javascript), а також контекстні об'єкти моделі, контролера та запиту.

Бізнес рівень (рівень бізнес-логіки) – це набір компонентів, відповідальних за обробку даних, отриманих від рівня представлення, реалізацію всієї необхідної логіки програми, здійснення всіх обчислень, взаємодію з базою даних та передачу результатів обробки на рівень презентації.

Шар доступу до даних – це такий, що містить моделі, що описують використовувані об'єкти та конкретні класи, використовувані різними технологіями доступу до даних, такими як контекстні класи даних Entity Framework. Він також зберігає сховище для взаємодії шару бізнес-логіки з базою даних.

Варто пам’ятати, що крайні рівні не можуть спілкуватися один з одним. Це означає, що рівень презентації (на відміну від контролера ASP.NET MVC) не має прямого доступу до бази даних або рівня доступу до даних, а лише до рівня бізнес-логіки.

Зазвичай в системах не буває класичної трирівневої архітектури, оскільки не завжди вистачає саме трьох рівнів. Частіше використовується більша їх кількість. Концепція залишається незмінною, проте з додатковими правками, що необхідні для системи. Таким чином і з’являються додаткові модулі обробки даних. Загалом прикладні шари або ж логічні рівні розділяють згідно типів функціонування та іноді за рівнем абстракції. Також варто враховувати, що будь-яка система використовує додаткове програмне забезпечення або проміжне програмне забезпечення і даний проект не є винятком. Структурна схема даної системи зображена у додатку Б.

Система складається з декількох основних модулів або підсистем. Першим розглянемо підсистему взаємодії з даними. Тут існує два варіанти використання даної системи, проте вони обидва спрямовані на отримання даних із зовнішньої системи або з просто ззовні системи та подальшої передачі їх у систему (рис 4.2). На вхід підсистеми надходять дані у певному вигляді, а саме у вигляді сформованого повідомлення із своєю структурою в service bus queue. Дана підсистема відповідає за зчитування даних з черги з подальшою їх обробкою та передачі на підсистему обробки даних. Варто зазначити, що на даному етапі також відбувається первинна обробка вхідних даних, їх валідація та формування із даних повідомлення для системи обробки даних, а також повернення результату про обробку назад у систему зберігання повідомлень Azure Service Bus.

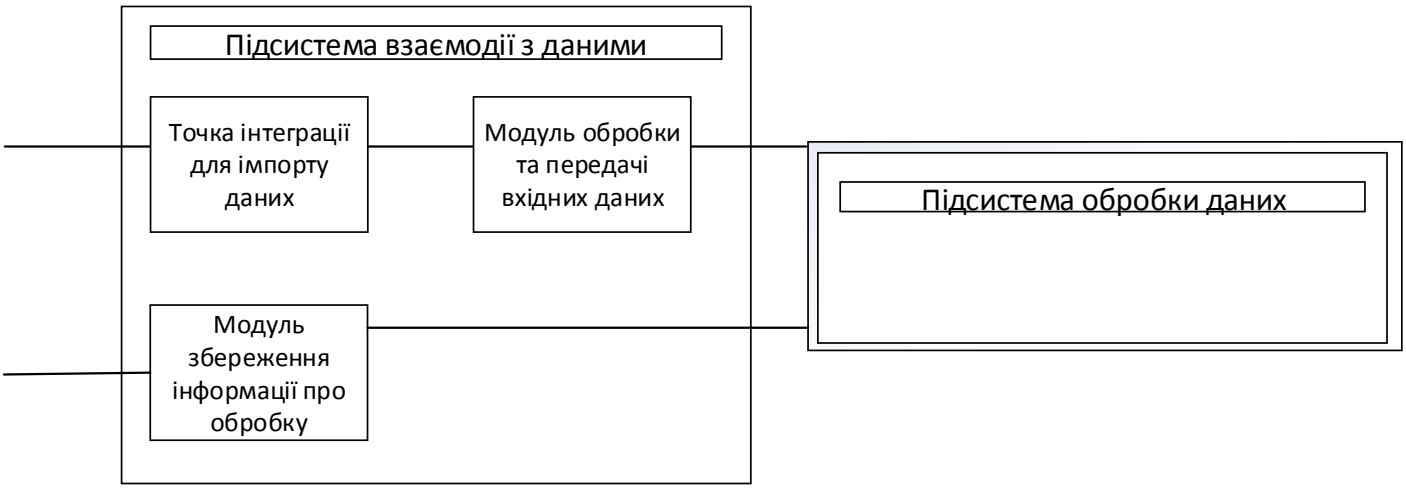

Рисунок 4.2 – Підсистема взаємодії з даними

Наступний рівень це рівень обробки даних в середині системи.(рис 4.3) Дана підсистема містить у собі три основних модулі. Це модуль обробки даних, модуль запису даних в систему та модуль збереження результату. Перший модуль представлений бібліотеками класів, або ж плагінами, які містять у собі реалізацію певної бізнес логіки. Даний модуль відповідає за попередню обробку даних. Всі процеси, що виконуються на цьому етапі відбуваються перед створенням або оновленням даних у системі, в той час, коли всі операції, що виконуються у модулі запису даних в систему навпаки відбуваються після збереження нових або після оновлення існуючих даних у системі. У даному модулі присутня логіка запису результатів обробки даних. Таким чином у системі створюється файл певного формату, який в собі містить корисну інформацію про всі зміни, що були створені в системі, а також передача результату обробки повідомлення з даними до підсистеми взаємодії з даними для подальшої обробки. Також даний модуль передає дані на подальшу обробку в підсистему зберігання даних безпосередньо у розроблюваній системі.

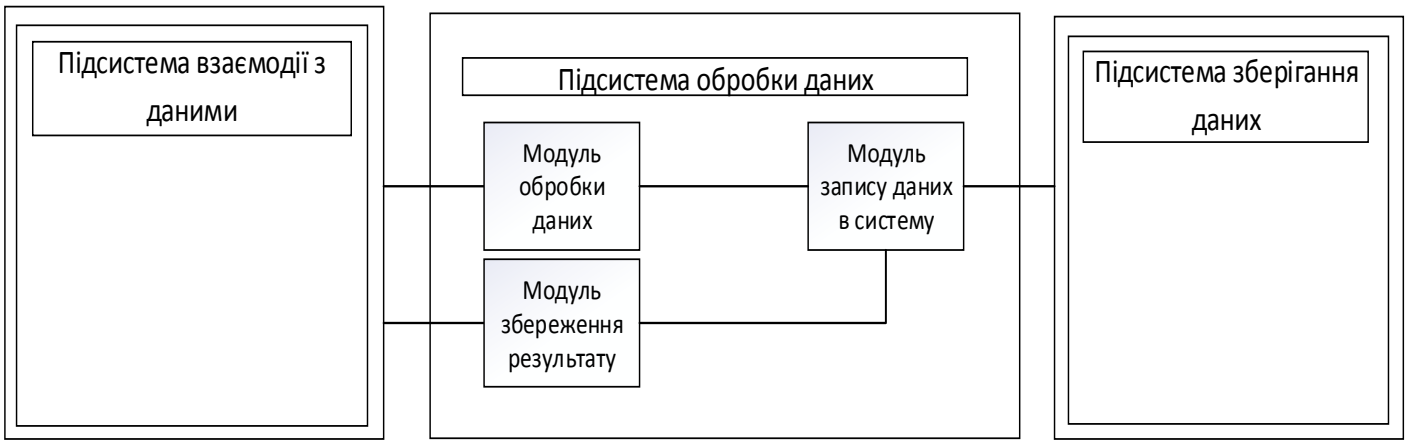

Рисунок 4.3 – Підсистема обробки даних

На наступному рівні розміщені сховища зберігання даних та метаданих. Ці сховища є табличними, а інформація до них поступає після фінальної валідації, яка може бути пустою на даному етапі. Отже, як можна побачити на рис. 4.4, ця підсистема складається з двох сховищ зберігання даних та модуля валідації.

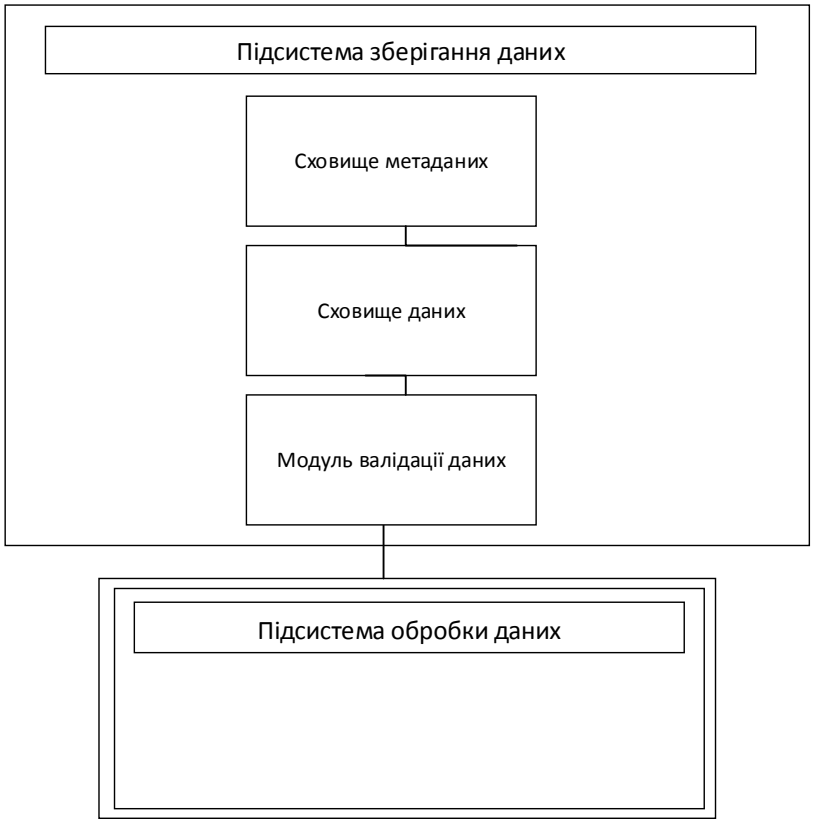

Рисунок 4.4 – Підсистема зберігання даних

Дві підсистеми, що залишились, логічно знаходяться на одному рівні. Ці підсистеми формування звітів(рис. 4.5) та відображення звітів(рис. 4.6). Більш детально розглянемо підсистему формування звітів. Вона складається з 4 основних модулів, серед яких: модуль запуску процесу формування звіту, модуль форматування звіту для збереження чи перегляду, модуль збору необхідних для звіту даних, модуль обробки даних для відображення звіту. На цьому рівні процес починається з модулю запуску формування звіту. Даний модуль може активуватись як періодично, за певним графіком, так і за вимогою користувача. У останнього варіанту також є два варіанти запуску. Перший передбачає завантаження користувачем системи середовища для перегляду звіту, що і є тригером для запуску даного модуля. Наступним кроком є модуль обробки даних для відображення звіту. На даному етапі робиться запит до системи з відповідною вибіркою заздалегідь заданою алгоритмом, а також можливо доповнений вибіркою фільтрів користувачем. Додатково даний модуль взаємодіє з модулем збору даних необхідних для звіту, до яких якраз належать налаштування користувача, обрані ним фільтри, мова тощо.

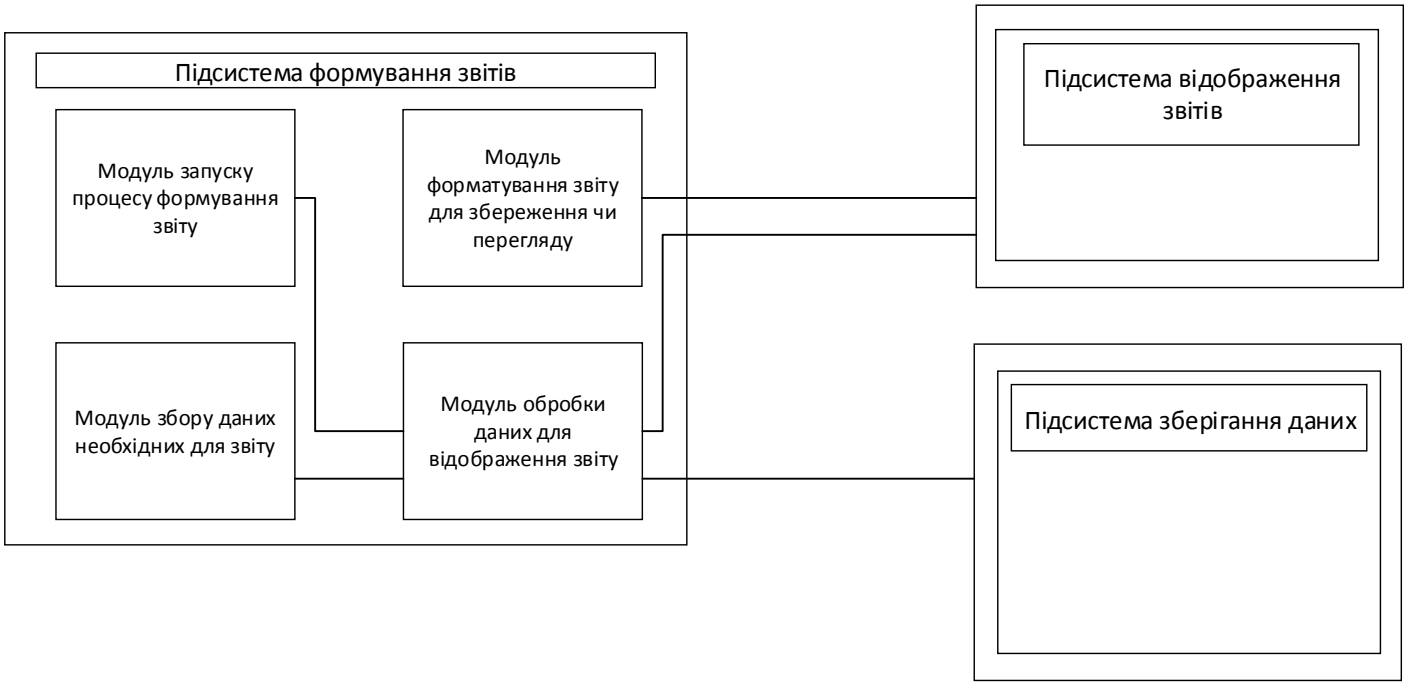

Рисунок 4.5 – Підсистема формування звітів

Після обробки даних, сам звіт передається у підсистему відображення звітів, в якій, в свою чергу, наявний модуль налаштування параметрів звіту, в якому користувач має змогу обрати формат файлу для перегляду, а також обрати тип перегляду. Після опрацювання даним модулем даних про звіт, вони надходять до модуля формування звіту для перегляду чи завантаження. На даному етапі звіт повністю сформований, у заздалегідь обраному форматі та або автоматично завантажується, або відображається у інтерфейсі користувача.

Підсистема відображення звітів складається всього із двох модулів, перший модуль відповідає за можливість надання користувачеві додаткових фільтрів та можливостей для налаштування кінцевого вигляду звіту, а другий відповідає безпосередньо за відображення у інтерфейсі користувача.

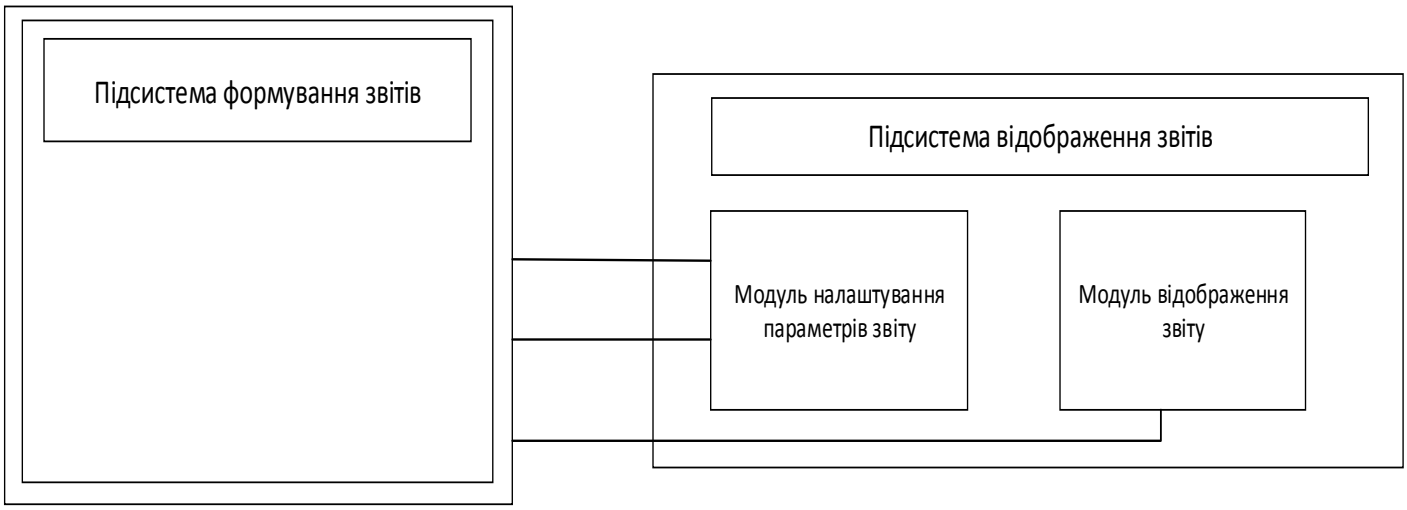

Рисунок 4.6 – Підсистема відображення звітів

Висновки до розділу

У даній системі існує необхідність впровадження багаторівневої архітектури через достатньо велику кількість логічних модулів, що функціонально знаходяться на різних рівнях. Даний підхід дозволяє чітко розподілити обмін даними між рівнями та передбачає, що рівні можуть обмінюватись даними лише зі суміжними. Така архітектура системи забезпечує можливість легкого розгортання системи, а також

відкриває можливості для масштабування системи за необхідності. При чому існує можливість модифікації як існуючих рівнів, так і створення нових рівнів у будь-якому місці системи без значних зусиль, зберігаючи програмні інтерфейси взаємодії.

5 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Визначення технологій, що будуть використані для проектування системи є дуже важливим питанням на етапі проектування системи. Хоч система має декілька рівнів, що дозволяє використовувати різні технології на різних рівнях, проте обрати технології для кожного етапу та чітко визначити увесь перелік. Оскільки зміна технологій на ходу суттєво збільшує час розробки системи та налаштування взаємодії із ними. Враховуючи те, що існують певні обмеження у використанні технологій в рамках певних систем, варто враховувати можливості програмних інтерфейсів взаємодії окремих компонентів та чітко сформулювати та описати їх ще на етапі проектування.

5.1 Вибір платформи розробки системи

Виходячи з визначених, у минулому розділі вимог до системи, платформою для розробки було обрана платформа .NET Core для компонентів системи, що відповідають

за інтеграцію та платформи .NET Framework для компонентів обробки даних всередині системи. Нижче детальніше описані дані платформи.

.NET Core - це платформа розробки загального призначення з відкритим кодом, якою керують спільноти Microsoft та GitHub .NET. Вона є кросплатформенним продуктом (підтримує Windows, macOS, Linux) і може використовуватися для створення програмних додатків, хмарних програм та IoT-програм. Загалом структурно ці дві платформи схожі, архітектура платформ зображена на рис.5.1.

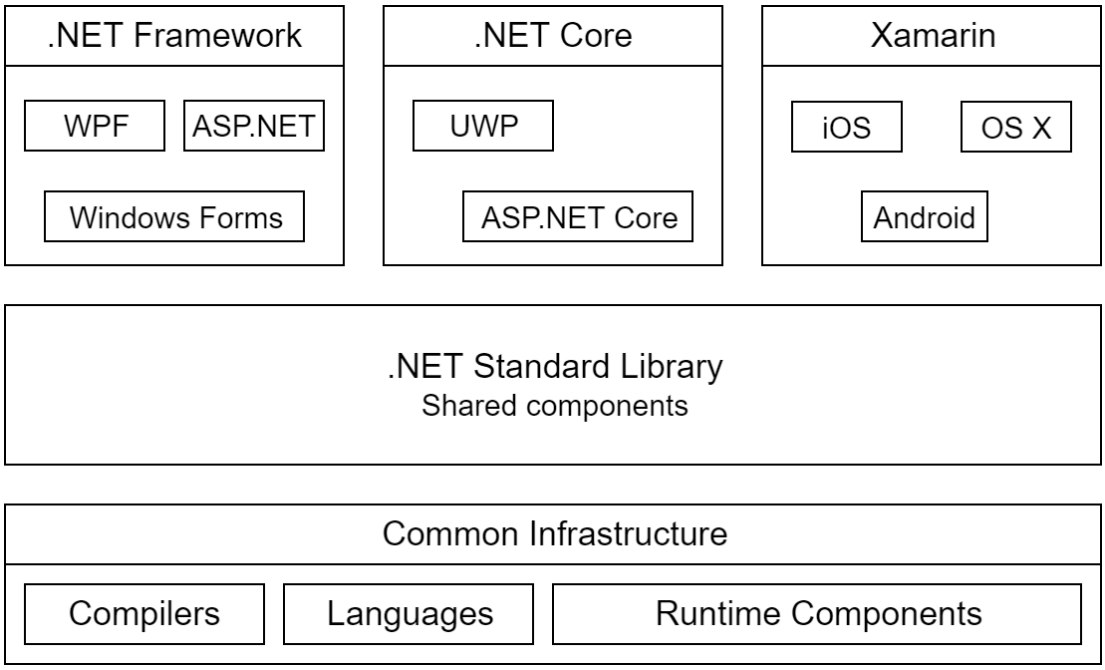

Рисунок 5.1 – Інфраструктура платформи .NET

Переваги .NET Core:

- якщо необхідно створити нову програму і є можливість вибрати між .NET Core та .NET Framework, .NET Core - найкращий вибір;

- Microsoft випустила .NET Core v 3.0, вдосконалену версію .NET Core. Даний продукт розвивається і скоріш за все, в майбутньому саме він і буде надалі розвиватись компанією Microsoft. .NET 3.0 тепер підтримує форми WPF та Windows. .NET Core 3.0 також підтримує перехресну розробку між формами UWP, WPF та Windows. Це дає

розробникам можливість впроваджувати новітні інтерфейси UWP у Windows Forms та WPF;

- .NET Core краще підходить для потреб платформ. Windows, Linux та macOS підтримують додатки .NET Core. Windows, Linux та macOS підтримують Visual Studio Code, популярний редактор відкритого коду Microsoft. Код VS підтримує сучасні потреби редакторів коду, такі як IntelliSense та налагодження. Більшість сторонніх редакторів (Sublime, Emacs, VI та ін.) Можуть використовувати .NET Core.

- .NET Core підтримує архітектуру мікросервісів, яка дозволяє використовувати міжплатформні послуги з .NET Core, включаючи використання .NET Framework, Java, Ruby або інших розроблених сервісів.

- контейнери - це сьогоднішні віртуальні машини. Модульність, невелика вага та гнучкість .NET Core дозволяє легко розгортати додатки .NET Core в контейнери. Контейнери можна розгорнути на будь-якому хмарному середовищі, Linux або Windows платформі. .NET Core добре працює як з сервісами Docker, так і з Azure Kubernetes.

Далі більш детально описано платформу .NET Framework. .NET Framework - середовище виконання програми для керування програмами, орієнтованими на .NET Framework. Він складається з загальної мови виконання, яка забезпечує управління пам’яттю та іншими системними послугами, та великої бібліотеки класів, яка дозволяє програмістам використовувати надійний та підтримуваний код у всіх основних сферах розробки додатків. .NET Framework - це середовище керованого виконання для Windows, яка може надавати різні послуги запущеним програмам. Складається з двох основних компонентів: Загальна мова виконання (CLR) (механізм виконання, який обробляє запущені програми) та бібліотека класів .NET Framework надають перевірену та багаторазову бібліотеку кодів, яку розробники можуть використовувати Телефонуйте цей код у власній програмі. Послуги, що надаються .NET Framework для запуску програм, включають:

- управління пам'яттю. У багатьох мовах програмування програміст відповідає за розподіл та звільнення пам'яті, а також за обробку життя об'єктів. Для програм .NET Framework CLR надає ці послуги від імені програми;

- загальноприйнята система. У традиційних мовах програмування базові типи визначаються компілятором, що ускладнює взаємодію між мовами. У .NET Framework основні типи визначаються системою типів .NET Framework і є загальними для всіх мов, на які спрямований .NET Framework;

- широка бібліотека класів. Програмісти можуть використовувати прості у використанні типи бібліотеки класів .NET Framework та її бібліотек-членів без необхідності писати багато коду для обробки звичайних операцій програмування низького рівня;

- рамки розвитку та технології. .NET Framework включає бібліотеки, призначені для домену, для розробки додатків, такі як ASP.NET для веб-додатків, ADO.NET для доступу до даних, Windows Communication Foundation для сервісно орієнтованих програм, Windows тощо;

- мовна сумісність. Компілятор мови .NET Framework видає проміжний код під назвою Загальний проміжний мова (CIL). Він складається під час виконання за допомогою загальної мови виконання. За допомогою цієї функції інші мови можуть отримати доступ до процедур, написаних однією мовою, і програміст зосереджується на створенні програм на бажаній мові;

- сумісність версій. За дуже невеликими винятками, програми, розроблені за допомогою конкретної версії .NET Framework, можуть працювати без змін на пізніших версіях;

- .NET Framework допомагає вирішувати конфлікти версій, дозволяючи на одному комп’ютері кілька версій загальної мови виконання. Це означає, що декілька версій програми можуть існувати одночасно, і програма може працювати на версії .NET Framework, побудованої разом з нею. Паралельне виконання працює з групами версій .NET Framework 1.0 / 1.1, 2.0 / 3.0 / 3.5 та 4 / 4.5.x / 4.6.x / 4.7.x / 4.8;

- кілька цілей. Орієнтуючись на .NET Standard, розробники можуть створювати бібліотеки класів, які працюють на декількох платформах .NET Framework, підтримуваних стандартною версією. Наприклад, націлювання на бібліотеку .NET Standard 2.0 може використовуватися в націленні на додатки .NET Framework 4.6.1, .NET Core 2.0 і UWP 10.0.16299;

- його обирають, якщо потрібно швидко створити та опублікувати щось, не вивчаючи .NET Core, .NET Framework - це вдалий вибір. .NET Core потребує тривалого навчання;

- також обирають .NET Framework, якщо є необхідність підтримувати та модернізувати існуючі програми .NET. При перенесення існуючої програми .NET до програми .NET Core існує необхідність у дописані певного коду та виконанні певної роботи;

- багато сучасних реалізованих систем та компонентів виконані з використанням саме .NET Framework;

- поточна версія 4.8 .NET Framework повинна бути останньою версією .NET Framework. Ніякої нової версії .NET Framework не планується. З одного боку це означає, що ніяких змін у нинішній версії вже не буде, а з іншого означає зупинку підтримки через певний час.

Також варто зазначити про особливість платформи, яка суттєво підвищує продуктивність навіть у порівнянні із нативними рішеннями. (див. таблицю 5.1) Дана особливість полягає в тому, що програмний код компілюється не в машинний, як це робиться у більшості мов програмування, а у специфічне визначене проміжне представлення. Дане представлення зберігає код програми у спрощеному вигляді, що структурно схоже на бінарне дерево. Її використання допомагає швидше використовувати зкомпільовані елементи коду. Насправді дане рішення дещо ускладнює реалізацію і у більшості задач підвищує продуктивність, але у специфічних ситуаціях, при специфічних задачах існують ризики отримати час на обробку значно довший, ніж класичний з трансформацією коду у машинний без проміжних етапів. Проте таких задач

достатньо мала кількість і такою затримкою можна знехтувати, оскільки позитивна сторона цього технологічного рішення значно більша, ніж негативна.

Також важливим фактором є те, що саме під дані платформи існують SDK під більшість хмарних середовищ, що спростить інтеграцію та розгортання з іншими сервісами у майбутньому. Звісно передбачити усі можливі варіанти розвитку подій неможливо в контексті швидкого розвитку програмних продуктів, проте наявність SDK та підтримки очевидно є суттєвим фактором при обранні стеку технологій для розробки. Тому дані платформи найкраще підходять для розробки нашої системи.

Таблиця 5.1 – Порівняння продуктивності веб-фреймворків для задачі видавання неформатованого тексту (дані сайту www.techempower.com [2])

Позиція	Фреймворк	Запитів у секунду	Порівняльний відсоток	Платформа
1	hyper	7,006,181	100.0%	Rust
2	tokio-minihttp	7,006,181	100.0%	Rust
3	ulib-plaintext_fit	7,004,608	100.0%	C++
4	actix	7,000,911	99.9%	Rust
5	ulib	6,998,172	99.9%	C++
6	libreactor	6,997,422	99.9%	C
7	actix-raw	6,996,104	99.8%	Rust
8	atreugo-prefork	6,995,436	99.8%	Go
9	firenio-http-lite	6,994,344	99.8%	Java
10	aspcore	6,993,704	99.8%	.NET
11	aspcore-rhtx	6,990,400	99.8%	.NET
12	wizzardo-http	6,987,612	99.7%	Java

13	fasthttp	6,983,911	99.7%	Go
14	may-minihttp	6,977,134	99.6%	Rust
15	fasthttp-prefork	6,969,608	99.5%	Go

5.2 Вибір основи для системи

Враховуючи те, що основною функцією нашої системи є впровадження алгоритмів зберігання та обробки даних та налаштування аналітики та алгоритмів підсумкового аналізу, було рішення, а саме CRM систему, яка мала би базовий функціонал, налаштувати алгоритми обробки даних, а також інтеграцію з існуючою системою, щоб система мала робочий вигляд, а також була доступна для користування та впровадження у бізнес, також зекономити час на розгортанні системи у хмарному середовищі, а також зекономити фінансові ресурси для цього. Водночас сильним вагомим плюсом при такому рішенні є економія часу на розробці та підтримці інтерфейсу користувача (див рис 5.2 та рис 5.3).

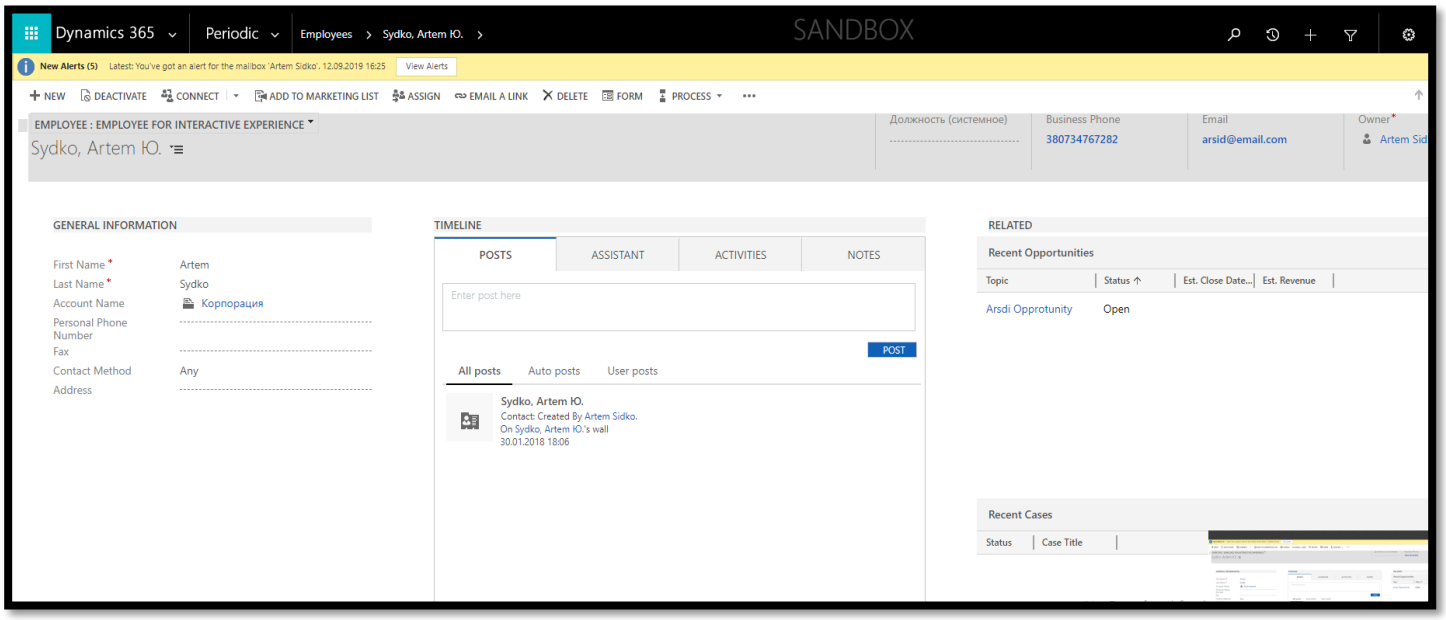

Рисунок 5.2 – Інтерфейс користувача у налаштованій системі Dynamics 365

Враховуючи вимоги до системи, а також вищеперераховане, в якості системи-основи для нашої системи було обрано CRM систему Dynamics 365 від компанії Microsoft. Дана система має чимало як переваг, так і недоліків. Нижче розглянуті основні із них:

- activity sorting control. У даній системі, як і у аналогів, розглянутих вище, наявні базові сутності з декількома типами activity. Екземпляри даної сутності спрямовані на зберігання інформації по активностям роботи з користувачем. Наприклад, телефонні дзвінки, зустрічі і т.д;

- cortana Integration. Також доступна інтеграція з системою персонального помічника на зразок Siri від Apple;

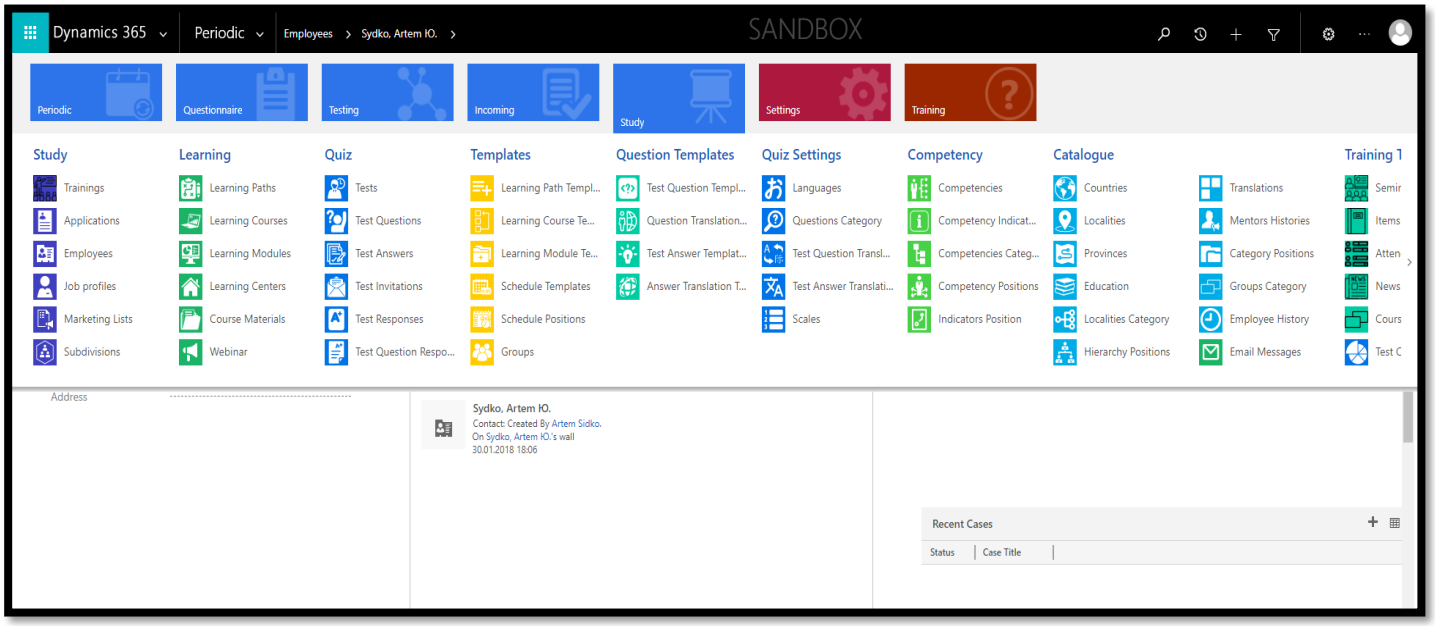

Рисунок 5.3 – Sitemap у налаштованій системі Dynamics 365

- customer insights. Наявний базовий функціонал, що надає доступ користувачеві для побудови графічного зображення аналітики;

- define access permission. У системі реалізовані налаштування ролей безпеки, що передбачає налаштування перегляду даних, доступ до внесення змін певних компонентів системи тощо;

- dynamics 365 app for Outlook. Наявний налаштований поштовий клієнт, що дає можливість формувати маркетингові розсилки не великих масштабів;

- process enhancements. Наявний інструмент для користувача системи, який надає можливість створювати автоматизовані бізнес процеси. Даний інструмент дуже сильно обмежений, тому дозволить користувачеві створювати лише базові процеси, тому розробка більш складних процесів покладена на програмний код;

- programmatic management. Система дуже гнучка для впровадження програмних модулів, таких як plug-in, вебресурси тощо. Це є дуже важливим фактором, оскільки дана система є лише скелетом, якого у стартовому вигляді не можна використовувати одразу, а потрібно доволі довго налаштовувати. З одного боку це спричиняє суттєві витрати часу, але водночас робить систему гнучкою та доступною для будь-якого користувача;

- server-to-server authentication. Наявна аутентифікація через Azure AD, також наявні налаштування аутентифікації для користувачів. Наразі доступно три типи: анонімна, двухфакторна, та типова;

- use form scripts. У системі є можливість налаштовувати зовнішній вигляд системи при необхідності, додавати усі необхідні для користувача операції, навіть є спеціальна SDK для розробки використовуючи мову програмування Java Script;

- client APIs. Система є гнучкою, вона орієнтована на інтеграції із сторонніми системами;

Варто зазначити, що особливості даної системи скорочують час простою ресурсів, а автоматичні сповіщення, що доступні для всіх пристроїв(є додаток для смартфону та планшетів) значно прискорюють надання послуг. Ця функція також дозволяє відслідковувати віддалені пристрої, і замовник завжди про це знає, тобто можна в режимі онлайн мати розуміння про доступність робочих ресурсів. Ви також можете використовувати програмне забезпечення для оптимізації планування ресурсів, доповнювати кодову базу для забезпечення необхідних кінцевому користувачеві

функцій. Цього можна досягти шляхом розробки конкретного рішення та автоматичного планування декількох проектів на регулярній основі. Це дозволяє більш ефективно використовувати заплановані ресурси, скорочуючи час продажів. Оптимізуючи свій ресурсний план, ви можете побачити перешкоди та цілі, такі як часові рамки та необхідні навички. Таким чином аналітика, що можна налаштувати у системі програмно надає суттєвих переваг над продуктами замінниками та конкурентами.

Резюмуючи, зазначимо, що дана система підходить як основа для впровадження та налаштування бази даних, обробки даних та відображення у зручній формі для користувача, а також автоматизації певних бізнес процесів та представленню аналітики у вигляді звітів користувачеві. В системі наявні базові функції та користувацький інтерфейс для базових процесів, що значно скорочує витрати часи на розробку нового та інтуїтивно зрозумілого вигляду системи.

5.3 Вибір хмарних сервісів для побудови системи

Серед багатьох доступних на сьогодні хмарних платформ була обрана платформа Microsoft Azure. Вона чимало переваг, але почати варто з того, що постачальником даної платформи також є корпорація Microsoft, яка передбачила варіанти інтеграції системи Dynamics 365 із зовнішніми системами шляхом використанням веб задач для цього та ще багато інших доступних технологій, серед яких:

- queue та topics. Даний компонент представляє собою впорядкований (FIFO) канал передачі повідомлень. Відмінність між двома цими компонентами полягає у тому, що черга це окремий елемент із своїм налаштуванням, а розділ поєднує у собі декілька черг, які можуть мати кожна свого користувача (див рис 5.4);
- web jobs. Даний компонент представляє із себе функцію, що використовує платформу .NET Core та використовує SDK для роботи з MS Azure. Дана функція може бути, як у стані Continues, так і у стані Triggered. У першому стані веб задача підключається до черги через строки підключення та очікує надходження повідомлення.

Одразу після надходження, функція запускається та оброблює дане повідомлення і далі переходить у режим очікування. У другому стані функція очікує зовнішнього запуску або ж існує можливість налаштувати scheduler, який запускатиме функцію у певний час або з певним проміжком часу. Для нашої системи ми будемо використовувати Continues Web Job, оскільки в чергу окремим повідомленням будуть надходити дані про розпізнаний об’єкт, а наша функція одразу оброблюватиме ці дані;

- app service. Представляє із себе віртуальний сервер, в якому можна розміщувати веб застосунки, також містить у собі блок конфігурацій та може містити черги та веб задачі;

- application insights. Інструмент отримання та відображення аналітики, що надається сервісами Azure. Даний інструмент є дуже корисним та зручним у використанні, коли потрібно отримати підсумкові результати відпрацювання веб додатків або ж інформацію про помилки, якщо такі були наявні у процесі виконання або обробки вхідних повідомлень.

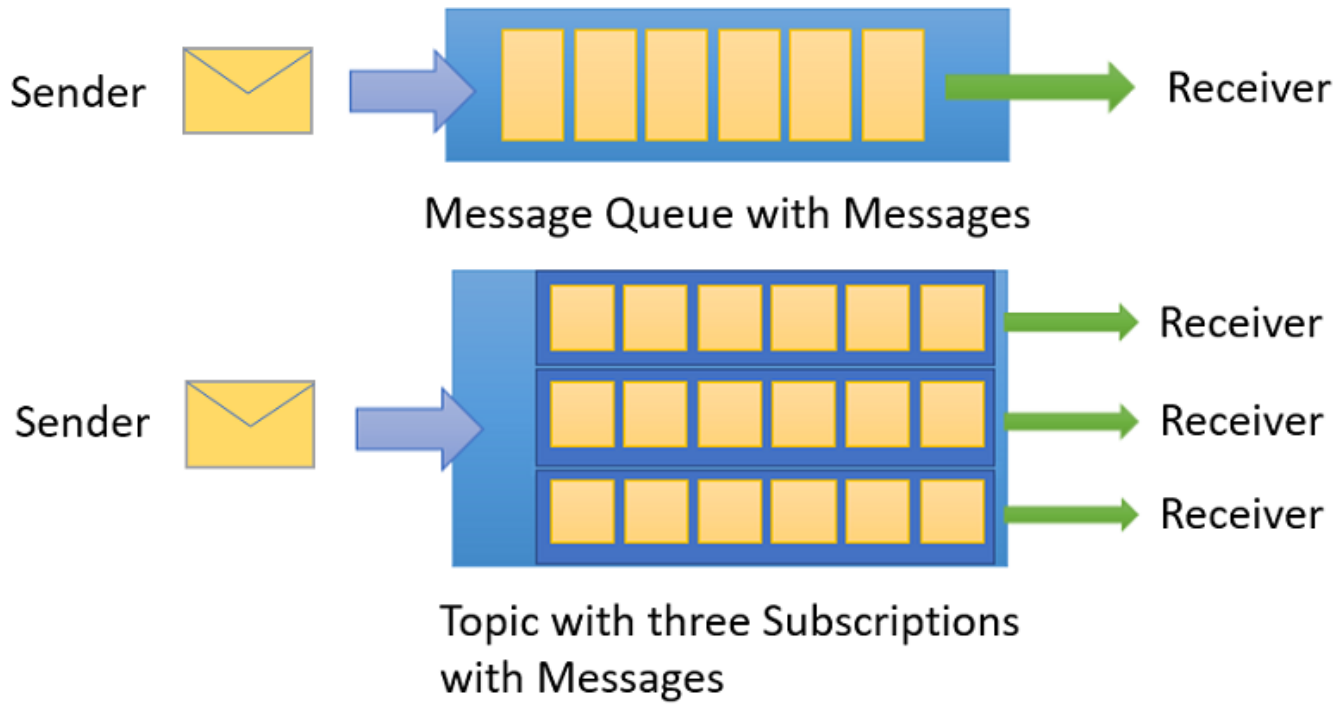

Рисунок 5.4 – Відмінність Queue від Topic

Детальніше варто зупинитись на App Service Plan. Враховуючи те, що постало завдання інтегруватись із системою розпізнавання образів, варто зазначити, що це буде зручніше зробити, якщо веб задача і зовнішня система будуть межах одного App Service. Ця умова не є обов'язковою, проте може значно полегшити процес інтеграції. Саме тому було прийняте рішення про розгортання ресурсів саме в межах одного плану. В такому варіанті реалізації веб-застосунки будуть працювати в межах одного віртуального серверу. При використані даного типу ресурсу є можливість швидкого масштабування, тобто збільшення таких віртуальних серверів або ж збільшення потужності обрахувальних ресурсів. Дані зміни не будуть безкоштовними, оскільки в основі змінення потужності лежить збільшення витрат на обслуговування даних ресурсів.

Також окремо варто зупинитись на використанні queue та topics. Оскільки вони не існують у системі як окремі об'єкти, визначимо, що існує певний об'єкт-сховище для них. Це Azure Service Bus. Даний об'єкт містить свої налаштування доступу, певні свої характеристики та опції, що можна змінювати, такі як послідовність повідомлень, аналітика по використанню ресурсів, налаштування обробки deadletters або повідомлень, що були оброблені у помилками тощо. Використання цього елементу дає можливість реалізувати паттерн «Брокер повідомлень» або Broker pattern, що являє собою архітектурну схему, яка може бути використана для побудови розподіленої програмної системи з ізольованими компонентами, які взаємодіють через виклики віддалених служб. Проксі-компонент координує комунікації, такі як запити переадресації, надсилання результатів та виключень.

5.4 Вибір технологій для аналітики

Аналітика – потужний інструмент обробки даних та виявлення вагомих даних, обробка їх та подання у певному вигляді. Алгоритми підсумкового аналізу полягають у певній сукупності дій при обробці даних для вирішення завдання моніторингу даних у

системі. Сховище даних, що використовуються у системі Dynamics 365 – це SQL база даних. Підсумковий аналіз передбачає виокремлення корисних даних із сховища даних та представлення їх у певному вигляді, зручному для сприйняття та розуміння. Для реалізації даного моніторингу необхідно робити запити до БД, використовуючи мову SQL(Structured Query Language). Дана мова є декларативною мовою програмування для створення, модифікації та управління даними у реляційних базах даних, що управляються відповідною системою управління базами даних. CRM система в даному контексті представляє своєрідну СУБД, оскільки має відповідні риси. Таким чином напряму з БД ми не маємо змоги взаємодіяти, проте є вихід у налаштуванні Reporting service.

5.4.1 Reporting Service

Reporting Service або SSRS – це серверна програмна система створення звітів, розроблена корпорацією Microsoft. Її можна використовувати для створення багатьох інтерактивних друкованих і не тільки звітів. Системою керують через веб-інтерфейс. Служба звітності реалізує інтерфейс веб-служб для підтримки розробки загальних програм звітування.

Користувачі можуть безпосередньо користуватися веб-службою сервера звітів або менеджером звітів (веб-додатком, який взаємодіє з веб-службою сервера звітів). Менеджер звітів дозволяє переглядати та керувати звітами, а також керувати та маніпулювати джерелами даних та налаштуваннями безпеки. Звіти можна надсилати електронною поштою або записувати до файлової системи як звичайні файли. Захист ґрунтується на ролі безпеки і може бути розшарованим на одному елементі, такому як звіт, джерело даних, каталог елементів або загальний веб-сайт. Ролі та дозволи безпеки є спадковими та можуть бути перероблені.

На додаток до використання окремого сервера звітів, який постачається разом із SQL сервером, також існує можливість переглядати звіти RDL за допомогою веб-

контролів ASP.NET ReportViewer або контролів за формами ReportViewer Windows Forms. Це дозволяє вставляти звіт безпосередньо у веб-сторінку або .NET-додаток. Контроль ReportViewer обробляє звіти одним із двох способів:

- на стороні сервера сервер звітів обробляє звіт;
- локальна обробка, де відповідне управління обробляє файл RDL незалежно;

Даний застосунок при коректному налаштуванні надає гнучкий дизайн, а також багато інструментів для обробки даних та виведення їх у зручному форматі. Для побудови запитів даних можна використовувати мови програмування SQL та fetchXml, остання з яких, це власна мова запитів до бази на основі мови розмітки XML. Було прийнято рішення використовувати сам цю мову при формуванні звітів, оскільки вона достатньо проста та інтуїтивно зрозуміла, а також схожа з методами отримання даних використовуючи SDK.

Звіти, що формуються використовуючи мову FetchXml формується довше майже в два рази, ніж звіти, що формуються з використанням запитів, написаними мовою SQL. Проте така ситуація лише із звітами, що формуються вперше. Надалі запити, що часто формуються зберігаються у пам’яті та часто можуть видавати результати швидше. В усьому іншому запити аналогічні та структурно схема не змінюється в залежності від використання того, чи іншого способу написання. Схема роботи Reporting Service зображена на рис. 5.5

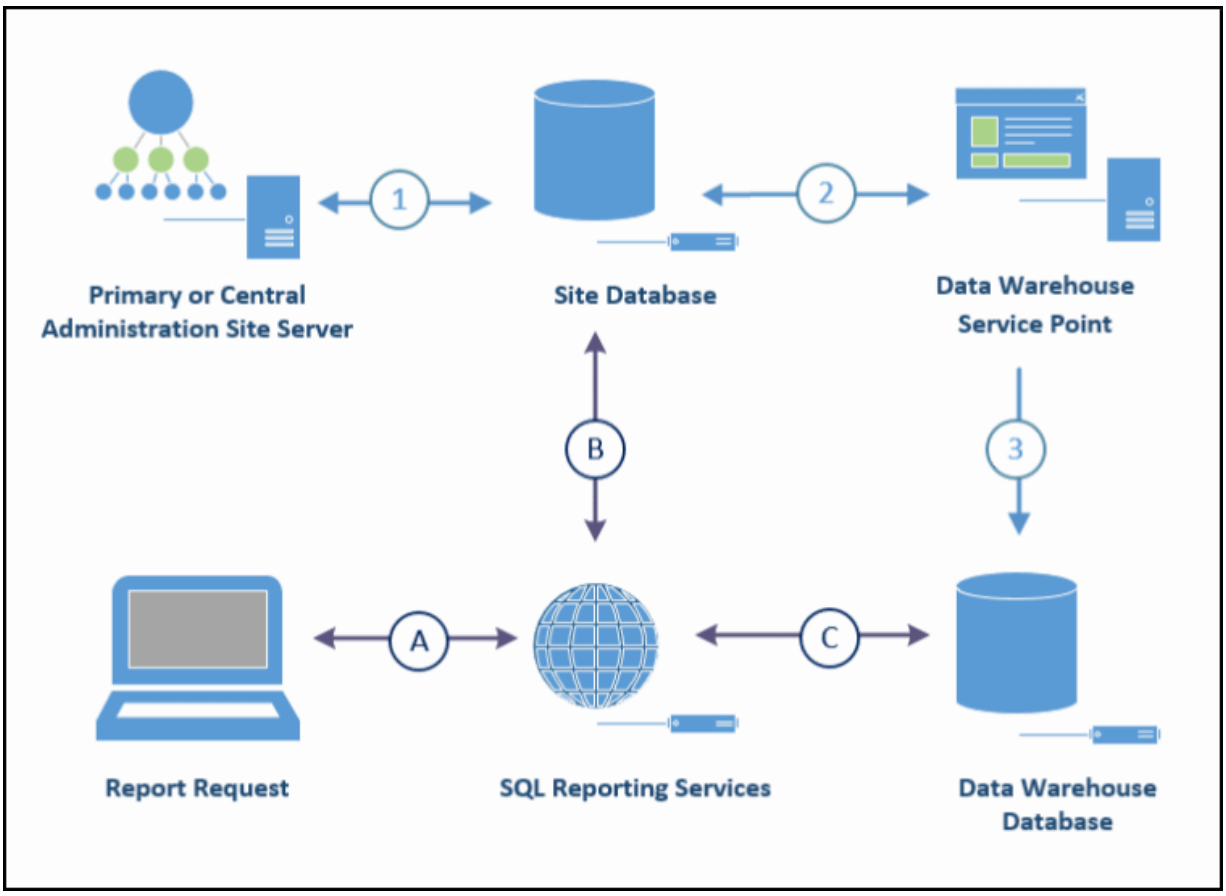

Рисунок 5.5 – Схема роботи SQL Reporting Service

5.4.2 Звіти в Dynamics 365 CRM

В Dynamics 365 представлено веб-інтерфейс для перегляду звітів. В даному інтерфейсі окрім перегляду також наявна можливість для скачування звітів у декількох форматах, таких як pdf, word, excel, txt, rdl. Стандартно звіти у системі формуються якраз у форматі rdl, проте окрім веб-інтерфейсу, представленого в системі ніяк інакше його вміст не переглянути, тому даний формат найчастіше використовується лише для розробки та для попереднього перегляду. Також для розробки звітів для формування відображення та умов відображення даних використовується мова програмування Visual Basic. Дана технологія дещо застаріла і її місце поступово займає PowerBI, проте на сьогоднішній день формування звітів за допомогою SSRS є безкоштовною альтернативою вищезазначеного аналогу. Загалом у SSRS доступні всі ті самі

компоненти, що і в PowerBI, проте їх налаштування займає більше часу, а також процес впровадження є доволі затратним по часу.

5.5 Вибір додаткових технологій

Серед додаткових технологій, що необхідні для формування системи є перелік мов та технологій використання. Серед таких мов є Java Script, Html, Css. Дані мови використовуються для налаштування певних процесів на стороні клієнта, тобто такі, що виконуються безпосередньо у браузері споживача. Даний стек технологій не є новим та широко використовується у веб-додатках різної складності та різного профілю.

Java Script – мультипарадигменна мова програмування. Підтримує об'єктно-орієнтований, імперативний та функціональний стилі. Це реалізація мови ECMAScript (стандарт ECMA-262). JavaScript часто використовується як вбудована мова для програмного доступу до об’єктів програми. Найбільш широко використовувана мова сценаріїв у браузерах, що робить веб-сторінки інтерактивними. Основні архітектурні особливості: динамічне введення тексту, слабке введення тексту, автоматичне управління пам’яттю, прототипування та функції як об’єкти першого класу. На JavaScript впливає багато мов, метою під час розробки було зробити мову схожою на Java, але полегшити використання непрограмістів. Немає компаній чи організацій, які використовують JavaScript. JavaScript відрізняється від декількох мов програмування, використовуваних у веб-розробці. JavaScript є зареєстрованою в США торговою маркою корпорації Oracle.

HTML – стандартна мова розмітки для документів у всесвітній павутині. Більшість веб-сторінок містять описи розмітки HTML. HTML інтерпретується веб-переглядачем, а отриманий відформатований текст відображається на екрані комп’ютера чи мобільного пристрою. До версії 5 HTML був визначений як додаток SGML (відповідає стандартній універсальній мові розмітки ISO 8879). Специфікація

HTML5 заснована на DOM (Document Object Model). У всесвітній мережі Інтернет сторінки HTML зазвичай надсилаються з сервера до браузера через HTTP або HTTPS, використовуючи звичайний текст або шифрування. Протокол передачі HTTPS є достатньо захищеним для сьогоднішнього рівня технологій.

CSS – це формальна мова опису зовнішнього вигляду документа, написаного з використанням мови розмітки. Автори веб-сторінок використовують CSS для визначення кольорів, шрифтів, компонування окремих блоків та інших аспектів відображення цих сторінок. Основними цілями розробки CSS є опис логічної структури веб-сторінки (згенерованої за допомогою HTML або іншої мови розмітки) та опис зовнішності веб-сторінки (наразі генерується за допомогою офіційної мови CSS) Цей поділ збільшує зручність використання документа, збільшує гнучкість та контроль викладу, зменшує складність та відтворюваність структурного змісту. Крім того, ви можете використовувати CSS для візуалізації одного і того ж документа в різних стилях або методах виводу, включаючи презентації на екрані, презентації презентацій, промову в мовленні (за допомогою спеціального голосового браузера або екранного зчитувача) та виведення даних за допомогою брайлівського пристрою.

Сильно спростивши, можна зазначити, що HTML відповідає за структуру веб-інтерфейсу, JS за поведінку, а CSS за оформлення. Використання цих технологій достатнє для забезпечення робочого компоненту веб сторінки або ж частини інтерфейсу користувача.

Висновки до розділу

Виходячи з умов, що сформовані до кінцевої системи, були визначені компоненти взаємодії системи з іншими, а також у середині системи.

Таким чином для забезпечення інтеграції системи з зовнішньою системою розпізнавання образів буде використано Web Job на платформі .NET Core, розгорнутий на віртуальному сервері App Service в середовищі Azure. Також разом із ним буде

налаштована queue в тому ж віртуальному сервері в межах Azure ServiceBus та відповідно налаштовані ключі підключення до них.

Для обробки даних в системі будуть написані, скомпільовані та опубліковані в системі plug-ins, що представляють собою бібліотеку на платформі .NET Framework. Дані плагіни будуть спрацьовувати на підписані події створення\оновлення чи видалення екземплярів сутностей в системі, а також будуть розподілені на pre та post події.

Для забезпечення інтерфейсу користувача обрано стек технологій для веб розробки HTML, CSS, JS. Варто, що для розробки кросс-браузерного додатку необхідно використовувати функції, що підтримуються у всіх браузерах, що також нерідко є складною задачею. Проте враховуючи, що у нас інтерфейс користувача в більшій мірі використовується вже наявний у системі, а налаштування компонентів буде не у великій кількості, кросс-платформність додатку збережеться на високому рівні.

Для формування системи аналітики є необхідним налаштування SQL Server Reporting Service(SSRS) для можливості отримання даних з бази та формування звітів. Для налаштування логіки у формуванні самих звітів використовуватиметься мова Visual Basic для нескладних операцій та математичних калькуляцій. Також для формування вибірки даних(DataSet) буде використовуватись XML подібна мова формування запитів fetchXML.

6 ER-ДІАГРАМА

В попередніх розділах, враховуючи вимоги до системи, аналіз існуючих рішень та технології, доступні для використання, була обрана система Dynamics 365 в якості основи для розгортання системи.

В даній системі використовується база даних MS Sql. І хоча система не надає можливості створювати таблиці використовуючи sql запити, вона надає для цього інші інструменти, а саме: налаштування через інтерфейс користувача, а також механізм завантаження елементів бази даних, таких як таблиці та їх полей, шляхом завантаження рішень системи. Про це детальніше буде описано у наступному розділі, в якому будуть поетапно розглянуті шляхи завантаження та вивантаження рішення в системі. Коротко можна описати даний процес наступним чином. При створенні таблиці у базі даних, вона додається у певному стандартизованому вигляді у файлі метаданих рішення, що доступне для переносу. Відповідно у цьому ж місці зберігається інформація про наявні у таблиці атрибути, їх типи, а також взаємозв’язки з іншими таблицями та їх атрибутами.

Варто зазначити, що у даній системі не можливим створення зв’язків типу один до одного, а типи зв’язків багато до багатьох формуються автоматично незалежно від користувача та містять лише два атрибути, посилання на екземпляр одної сутності та іншої. Даний підхід має багато переваг, проте у випадку, якщо необхідно зберігати ще якісь атрибути у таблиці, що описує даний зв’язок, то це можливо зробити програмними засобами або ж внесенням змін безпосередньо у файл метаданих рішення, проте останній варіант вважається поганою практикою та може призвести до неочікуваних результатів роботи системи.

Також варто зазначити, що середовище знаходиться у хмарі, а отже доступу у режимі офлайн не надається, але так само є інструменти, що дозволяють робити локальну копію частини даних, яка необхідна для постійного доступу, в тому числі у режимі офлайн. Подібне рішення є зручним, проте потенційно може спричинити

надмірне навантаження на пристрій використання через об’ємну базу даних, тому даний інструмент потрібно використовувати дуже обережно.

Враховуючи визначені для системи сценарії використання, було встановлені наступні сутності:

- User;
- ImageData;
- Company;
- FilterCriteria;
- PracticeRecord;
- Contact;
- Subdivision;
- JobProfile;
- IntegrationSystem.

Сутність User містить обліковий запис користувача системи. Вона використовується для зберігання даних про користувача системи. Необхідні атрибути:

- securityRole;
- username;
- password;
- lastLogin;
- lastSessionTime;
- totalTime.

Дана сутність з усіма необхідними атрибутами наявна у системі та називається systemuser. Щоб не нагромаджувати систему зайвими елементами, які не матимуть потреби у існуванні через їхнє не використання. Одразу розглянемо другу сутність, яка аналогічно представлена у системі та має усі необхідні атрибути для використання відповідно до розроблюваної системи. Це сутність Contact. Її необхідні для системи атрибути перераховані нижче:

- firstname;

- secondname;
- phonenumber1;
- phonenumber2;
- emailaddress1;
- emailaddress2;
- subdivisionId;
- userId;

Дана сутність призначена для зберігання даних про фізичну особу користувача, має посилання на підрозділ, де працює користувач, займана позиція, а також контактні дані. Додатково сутність використовується у формуванні аналітики щодо ефективності роботи користувачів із системою, кількість запитів від конкретних користувачів, швидкість обробки даних тощо.

Що стосується додаткових сутностей варто зазначити, що їх створюємо використовуючи користувацький інтерфейс системи(рис 6.1). Даний інтерфейс дозволяє швидко налаштувати необхідні компоненти та зв’язки в середині системи між сутностями.

Одною з основних сутностей є сутність ImageData. Нижче наведено список її атрибутів:

- imageUrl;
- name;
- createdOn;
- createdBy;
- resultTypecode;
- subdivisionId.

В системі існують артибути базові для всіх сутностей такі як, createdOn та createdBy. Перше поле ми візьмемо із базового функціоналу, проте друге існє необхдність створити власноруч, оскільки стандартне поле посилається на екземпляр сутності systemuser, в той час, коли згідно вимог нам необхідно мати дані про фізичну

особу, яку презентує сутність Contact в нашій системі. Частіше за все у одного контакту буде один користувач(systemuser), проте існують варіант, при якому від одного користувача будуть мати доступ до системи різні фізичні особи. Це може бути допущено для здешевлення вартості підтримки системи, оскільки більша частина вартості – це якраз підписки користувачів, а вже потім використання даних системи.

Наприклад, дане використання можливо для випадку, коли працівник торгової точки певного магазину працює позмінно. Тобто дві або більше фізичних осіб можуть працювати використовуючи один обліковий запис, не створюючи конфліктів.

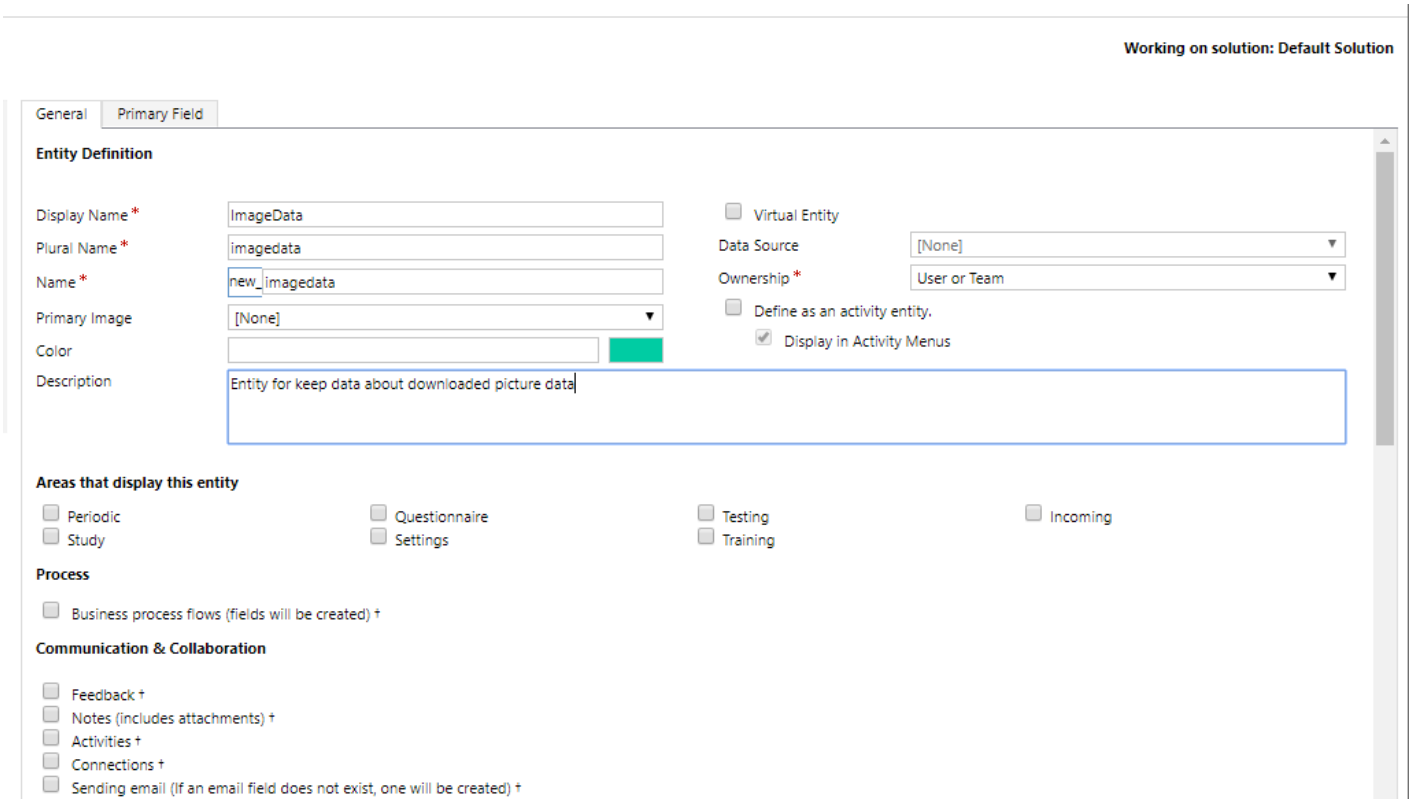

Рисунок 6.1. Користувацький інтерфейс для створення сутності в системі dynamics 365

Також у даної сутності є ще поле типу OptionSet. Це специфічним тип даних, наявний у системі Dynamics 365, що містить у собі словник значень типу int з відповідними текстовими значеннями. Це поле буде використовуватись для фіксації результату обробки зображення.

Наступна сутність Company. Дана сутність носить описовий характер та містить відомості про організацію або про клієнта, з яким співпрацює користувач системи. У сутності наявний такий перелік атрибутів:

- companyTypecode;
- name;
- workingFrom;
- primaryContactId;
- primaryEmail;
- subscriptionType.

У сутності наявні як і звичайні інформативні атрибути, такі як тип організації типу optionset, поля workingFrom типу dateTime, ключа зв’язаного контакту з даною організацією, через якого здійснюється комунікація щодо будь-яких питань(іншими словами, це представник компанії), так і поле, що визначає рівень підтримки, що надається даній компанії. Для прикладу можна розглянути три основні тип підписки: базова, стандартна та покращена. Базова підписка надає можливість завантажувати по одному файлу для кожної торгової точки в день. Стандартна підвищує цю кількість до 5 зображень на день, а також додає можливість автоматичного формування звіту. Покращена надає користувачам доступ до безлімітного завантаження зображень, а також до алгоритмів налаштування таймерів для запуску синхронізацією та автоматичною відправкою та формуванням звітів на електронну пошту.

Наступна сутність системи FilterCriteria. Дана сутність наявна як додатковий функціонал для користувачів, які вміють користуватись інструментами написання запитів використовуючи мову fetchXml. На практиці даний інструмент іноді надає додаткові можливості та пришвидшує роботу та ефективність взаємодії з системою. Дана сутність має наступний перелік атрибутів:

- createdBy;
- fetchXmlCode;
- type

- name.

Аналогічно необхідно перевизначити поле createdBy, оскільки воно використовує не той взаємозв’язок, що нам необхідний для формування даних. Поле fetchXml міститиме дані про код запиту чи фільтру, який користувач може додати вписавши код у це поле та відповідно поле type, що даватиме розуміння запит , чи лише фільтр зазначено в попередньому полі.

Наступна сутність це Subdivision. Дана сутність створена для того, щоб мати інформацію про конкретну торгову точку, а також про працівників, що там працюють та користуються системою. Також дана сутність бере участь у аналітиці, оскільки по ній формуються звіти та аналітика по торговій точці.

До списку атрибутів цієї сутності належить наступник список полей:

- name;
- address;
- primaryContactId;
- employeesCount;
- managerPhoneNumber;
- defaultAmountForOrder;

Стандартні інформаційні поля, що ми вже розглянули в рамках інших сутностей не будемо переглядати знову. Поле employeesCount призначене для зберігання кількості робітників, що працює на торговій точці більше для ведення аналітики, аніж для інших процесів.

Поле managerPhoneNumber містить у собі номер телефону менеджера відповідного за поставки товарів. Оскільки на основі оброблених даних формуються списки на поставки.

Поле defaultAmountForOrder створене для створення замовлень певної заздалегідь заданої кількості. Дане поле не є необхідним та його використання навіть не завжди є доцільним. Дане поле може бути використаним, якщо поставка по даній торговій точці відбувається по конкретній одиниці товару, що має чітко визначену

кількість та не має особливих специфікацій. Проте дане поле при налаштуванні під конкретний товар можна буде ефективно використовувати за замовчуванням. В такому випадку налаштовується автоматична поставка товару у разі його нехватки на торговій точці чи об’єкті. Також важливо розуміти, що у будь-який момент можна перестати враховувати дане поле при створенні замовлення.

Наступна сутність JobProfile і вона має наступний перелік атрибутів:

- name
- timeInMonths
- isHasAccess

Список полей на даній сутності маленький, проте містить логічну інформацію. Поле name інформативне та не несе особливого логічного навантаження. Поле timeInMonths зберігає дані про час, який співробітник займає дану позицію. Дана інформація може бути використана в аналітиці, щоб зробити пошук по найбільш досвідченого працівника.

Останнє поле isHasAccess створене спеціально для налаштування доступу користувача до звітів та до системи відправлення зображень загалом. В системі наявний механізм призначення бізнес ролей, проте дане впровадження дозволяє зручно вимкнути доступ до системи одразу для цілої групи користувачів, що належать до однієї групи.

Висновки до розділу

В даному розділі описані основні сутності системи та їх атрибути. Загалом сформовано деякі додаткові умови та описані можливості системи для розширення системи. Частково були запозичені вже існуючі сутності системи, оскільки це є доцільним. Також описана структура бази даних у системі Dynamics 365. На основі даного розділу була сформована ER-діаграма системи(див. Додаток Г). Загалом сформованого списку сутностей та відповідних їм атрибутів цілком достатньо для

реалізації поставлених до системи вимог, немає надлишковості, що також може бути негативним фактором, як було визначено на етапі аналізу існуючих рішень.

Також важлим фактором було створення такої структурної схеми, щоб було можливо зручного формування звітів для системи та уникнення складних моментів при проектуванні запитів.

Слід зазначити, що у всіх сутностях не були зазначені їх первинні ключі(primary keys). У кожного вони за назвою дублюють назву та в кінці є приставка id(наприклад contacted, subdivisionid тощо).

7 РЕАЛІЗАЦІЯ ЛОГІКИ СИСТЕМИ

Після вибору стеку технологій, що використовуються та аналізу вимог до системи було визначено, що система має бути модульною. Для забезпечення працездатності системи необхідно налаштувати середовище для коректної та ефективної роботи системи. Окремо варто виділити такі етапи підготовки системи:

- налаштування середовища Dynamics 365;
- налаштування хмарних сервісів;
- налаштування модуля інтеграції з системою розпізнавання образів;
- розробка модуля взаємодії з даними системи;
- розробка модуля обробки даних;
- розробка модуля формування звітів;
- налаштування модуля відображення звітів;
- налаштування модуля зберігання даних;
- розміщення модулей, що відповідають за виконання бізнес логіки у середовище dynamics 365 та у хмарне середовище;
- необхідне налаштування SSRS.

Налаштування системи в такій послідовності та по таким крокам дозволить розгорнути якісну робочу систему за мінімальні витрати часу.

7.1 Розгортання і налаштування хмарних сервісів

Для того, що зареєструвати instance Dynamics 365 необхідно мати обліковий запис Microsoft (можливо використовувати як особисті, так і робочі облікові записи) та активну підписку на Azure. Наступним кроком буде перейти у відповідний розділ з назвою Dynamics 365 на офіційному ресурі Microsoft та натиснути на кнопку «Отримати». Далі необхідно ввести дані про організацію чи підприємця, який використовуватиме систему та обрати необхідний функціонал системи(рис 7.1). Наразі

достатньо буде будь-якого із представлених варіантів, усі розроблені компоненти підтримують будь-яку версію додатку.

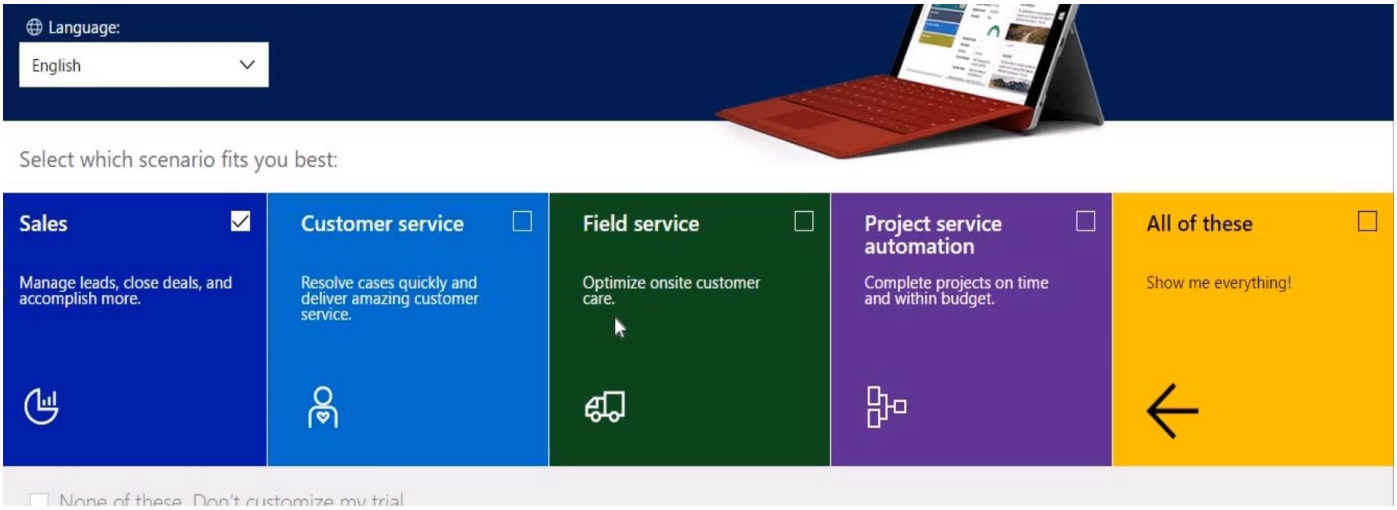

Рисунок 7.1– Вікно реєстрації інстансу Dynamics 365

При введенні всіх даних та налаштувань правильно, користувача буде перенаправлено на домашню сторінку системи. Якщо це сталось, система успішно створена. Наступним кроком буде завантаження рішення(solution), що містить усі необхідні компоненти системи, такі як плагіни, веб ресурси, звіти, налаштовані сутності, тощо. Щоб перенести таке рішення необхідно виконати наступні кроки:

- перейти до налаштувань та перейти до вкладки рішення;
- вибрати «Імпорт» у меню списку рішень;
- у діалоговому вікні "Імпорт рішення" перейдіть до стисненого файлу (.zip або .cab), який містить рішення, яке потрібно імпортувати, на кроці «Вибір пакета рішень»;
- є можливість переглянути інформацію про рішення перед тим, як вибрати імпорт;
- коли імпорт рішення завершиться, потрібно почекати деякий час. У разі успіху ви можете переглянути результати та вибрати пункт «Закрити»;
- якщо є необхідність у імпорті змін, які необхідно опублікувати, необхідно спочатку опублікувати налаштування, перш ніж буде можливість їхнього використання.

Якщо ж у користувача вже наявна підписка та розгорнута система, можна експортувати додаткові налаштування із неї, щоб додати у нову систему з налаштованими бізнес процесами. Для цього необхідно виконати наступні кроки:

- перейти до рішення налаштування;
- у списку вибрати рішення, яке потрібно експортувати, та вибрати пункт «Експорт»;
- на кроці користувацької публікації система повідомляє про експорт лише опублікованих налаштувань. Можна опублікувати всі налаштування, перш ніж вибрати «Далі»;
- якщо рішення потребує відсутніх компонентів, відображається крок "Відсутні необхідні компоненти". Можна ігнорувати це попередження, лише якщо імпортувати його назад у початкову організацію як некероване рішення. В іншому випадку необхідно дотримуватись інструкцій у діалоговому вікні, щоб скасувати експорт та додати необхідні компоненти;
- на кроці «Експорт параметрів системи (Додатково)» існує можливість обрання конкретних системних налаштувань, які потрібно включити до рішення. Якщо рішення залежить від набору системних налаштувань, необхідно обрати їх та натиснути «Далі»;
- докладніше про налаштування, що входять до кожної опції, можна переглянути у розділі «Налаштування параметрів»;
- на кроці типу пакета потрібно вибрати, чи потрібно експортувати рішення як некероване рішення або кероване рішення;
- потім можна вибрати цільове рішення для конкретної версії Dynamics 365. Ця опція зазвичай використовується ISV, які хочуть експортувати рішення, сумісні з попередніми версіями. Прийміть налаштування за замовчуванням, якщо не планується імпортувати це рішення в організацію, яка не була оновлена до тієї самої версії, яка використовується наразі;
- необхідно натиснути «Експорт», щоб завантажити файл рішення;
- точна поведінка при завантаженні файлу варіюється від браузера до браузера.

Перейдемо до налаштування хмарних сервісів. По-перше, корисно створити окрему групу ресурсів для проекту. Групи ресурсів дозволяють логічно групувати ресурси Azure. Створюючи більшість ресурсів, потрібно вказати регіон. Цей атрибут вказує центр обробки даних, в якому будуть розміщуватися дані. Регіони також можуть впливати на перелік доступних послуг та їх витрати. Всі наступні ресурси. Ми вибрали «Західну Європу», оскільки центр обробки даних є найближчим до області просування проекту.

Спочатку потрібно створити план обслуговування додатків у новоствореній групі ресурсів. Віртуальні сервіси використовуються як час виконання для системних модулів на основі функцій Azure. Під час створення можна вказати операційну систему (Linux або Windows). Ми вибрали цю операційну систему, оскільки план додатків служби Windows має більше функцій. Також треба вибрати рівень ресурсу. Для забезпечення безперебійної роботи системи слід вибирати принаймні стандартний рівень S1. Менші рівні не підтримують певні режими комутації.

Для зв'язку між функціями потрібно створити брокера повідомлень служб шини Azure. Завдяки структурній схемі зрозуміло, які функції взаємодіють. Правильне використання черг та методів теми та підписки є важливим. Потрібно проаналізувати систему та переконатися, що існує сувора взаємодія лише між двома модулями. Окрім веб-порталу, також зручно керувати ресурсами службової шини Azure за допомогою спеціального провідника службових шин. Приклад створення та конфігурації служби показаний на малюнку 7.2. Додаток Service Bus Explorer є дуже зручною альтернативою стандартним можливостям порталу, оскільки має багатший функціонал та можливості напряму зчитувати або видаляти повідомлення, а також отримувати інформацію щодо причин не відпрацювання алгоритму.

Створення та налаштування ресурсів Insights Application простіше, ніж інші ресурси. Для підключення до створеного екземпляра використовуйте InstrumentationKey. Це унікальний GUID (глобальний унікальний ідентифікатор), достатній для підключення програми до APM. Функціональні з'єднання включають

налаштування APPINSIGHTS_INSTRUMENTATIONKEY на панелі «Налаштування програми» для функціональних програм.

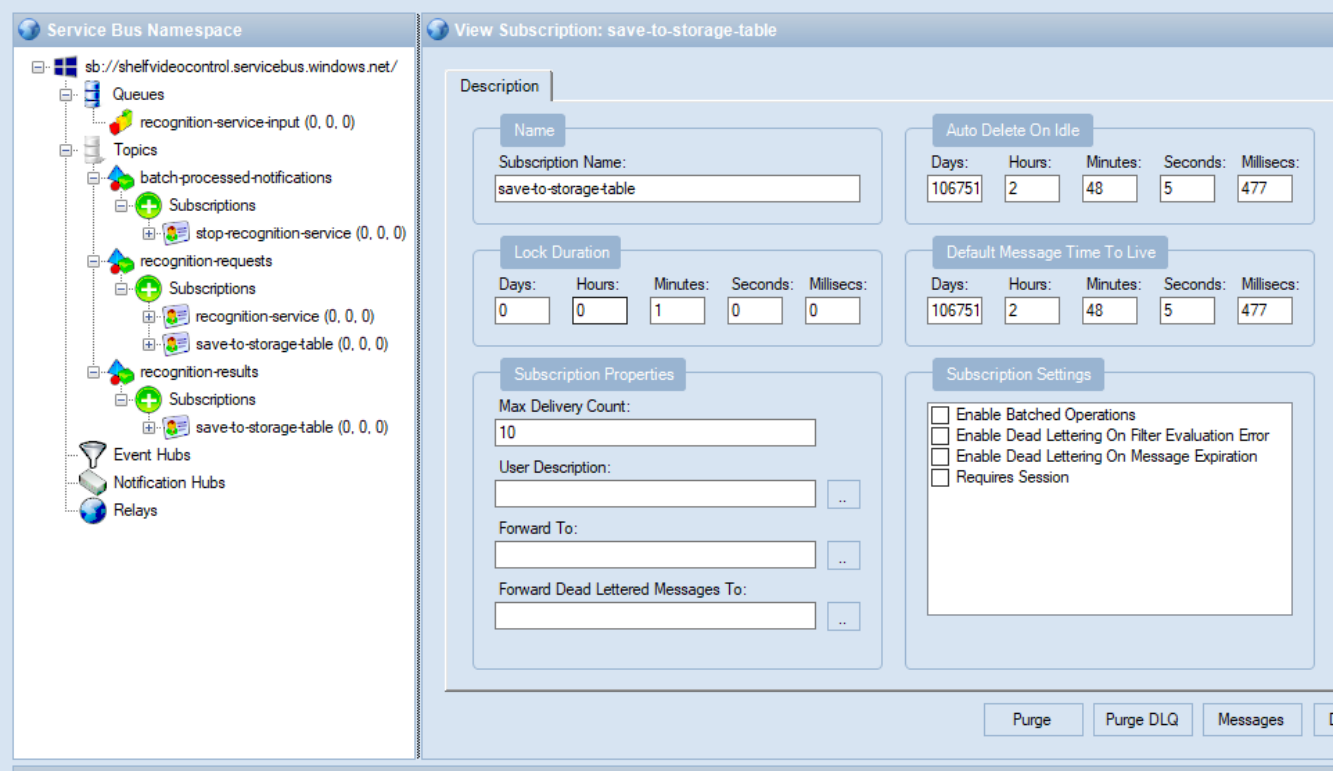

Рисунок 7.2 – Налаштований екземпляр сервісу у додатку Service Bus Explorer

a. Розгортання модуля інтеграції з системою розпізнавання образів

Даний модуль безпосередньо відповідає за логічну взаємодію системи на базі CPM та системи розпізнавання образів. В таблиці 7.1 описана структура даного модуля. Загалом він має 5 вхідних параметрів, та 2 вихідні параметри. Серед вхідних параметрів налаштування 3 визначають характеристику модуля, 2 ж описують логіку обробки даних. Два вихідні параметри дають зворотній зв’язок про результати відпрацювання як в CPM систему, так і в Azure Service Bus Queue.

Таблиця 7.1 – Структура веб задачі CrmJob

Назва задачі	CrmJob
Тригер	Повідомлення в AzureService Bus Queue
Залежності	Azure SDK Service Bus Queue Crm Sdk Dynamics 365 access
Параметри	Continuous – cron вираз регламенту запуску. ObjectsCountLimit – максимальна кількість об’єктів, що може опрацьовуватися за один раз. SourceDefinition – визначення джерела даних (типів оброблюваних повідомлень). Status – дані про теперішній статус роботи. AzureConnectionStrings – набір параметрів підключення до зовнішніх систем та черг повідомлень. RequestExpirationTime – параметр, відповідальний за час життя повідомлення, доступного для обробки.
Результат	StatusCode – дані щодо теперішнього статусу застосунку. RequestStatus – дані про запис даних в CPM систему.

Наступний модуль – це модуль формування звітів. Його структуру можна побачити на таблиці 7.2. У ній описані основні параметри, що впливають на результат відображення. Особливістю даного модуля є те, що він має два тригери для запуску і залежно від тригеру буде використовуватись та чи інша логіка виконання запиту до сховища даних. У структурі наявні 4 параметри, проте один з них, FilterCriteria, це комбінований параметр, що містить у собі додаткові параметри для формування звіту.

Таблиця 7.2 – Структура модуля формування звіту

Назва функції	GenerateReport
Тригер	Timer\ on-demand
Залежності	Dynamics 365 stable connection.
Параметри	FileType – тип файлу, в якому буде сформовано звіт. OpenMode – тип перегляду файлу. Даний параметр має два значення: перегляд у вікні браузера та скачування для перегляду. FilterCriteria – набір фільтрів для формування звіту. В залежності від типу запуску може бути два типи запитів. Для автоматичного запуску по таймеру використовується Sql запити, а для тих, що запускає користувач натисканням на кнопку або через веб-інтерфейс, використовуються запити з використанням мови запитів FetchXml. ReportId – GUID звіту, що необхідно запускати.
Результат	Даний модуль видає результат лише у випадку невдалого відпрацювання. Для невдалої обробки існує вивід повідомлення про помилку користувачеві. (Найчастіше помилки пов’язані з розробкою звіту та при його тестуванні. Таким чином кінцевий користувач побачить помилку, пов’язану зі звітам лише у випадку проблем із підключенням до середовища зберігання даних або ж обмеженням, викликаними налаштування середовища відображення або зберігання звітів.

Важливим модулем є модуль обробки даних. Блок-схема його роботи зображена на рисунку 7.3. На даному рисунку зображені основні компоненти алгоритму. Після обробки вхідних даних, що записуються у вигляді повідомлення системою розпізнавання образів, іде обробка даних та виконуються запити до системи, які

спрямовані на пошук сутності Company, до якої прив’язуються дані про оброблені системою зображення. Якщо екземпляра, вказаного у даних поки ще не існує в системі, він створюється з можливістю заповнення необхідних для подальшої обробки атрибутів. Якщо ж він наявний у системі, створюється додатково запис сутності ImageData, що зберігає в собі інформацію щодо оброблених зображень зовнішньою системою з необхідним набором даних. Аналогічний алгоритм відпрацьовую при обробці кожного нового зображення та передачі відповідного повідомлення до черги з боку зовнішньої системи.

Даний модуль дозволяє створювати записи у чітко структурованій формі, що у подальшому дозволить ефективно будувати звіти та аналітику щодо ефективності планування ресурсів, якості роботи персоналу, який відповідає за провадження матеріалу для обробки. Один із прикладів використання даного модуля є система магазинів, в яких мерчендайзер робить фотознімок полиці з товарами, система розпізнавання образів робить висновок щодо наявності необхідного товару на полиці та передає дані про того, хто завантажив знімок, дату, наявність, торгову точку, власника її тощо. Далі веб-задача зчитує повідомлення від системи розпізнавання образів і передає запит на створення відповідних даних у CRM систему. В системі спрацьовую модуль обробки даних та пророблює описані вище маніпуляції. Результатом роботи будуть створені в системі записи щодо наявності товару у торговій точці. Система звітів для такої системи дозволить швидко виявляти необхідність у поповненні асортименту товарів торгових точок. Також можливо налаштувати сповіщення шляхом відправлення електронного листа відповідному постачальнику про необхідність певної кількості товару на певній торговій точці.

Описаний приклад доступний до реалізації враховуючи наявні у системі модулі вже зараз. Проте дана система не обмежується цією сферою використання через її модульність, що позитивно впливає на масштабованість системи. Таким чином є можливість додання або навпаки видалення певних модулів взаємодії, при цьому не впливаючи на цілісність системи. Модулі є незалежними та працюють окремо один від

одного самостійно. Так само модуль формування звітів, що може налаштовуватись інструментами розробки SSRS Reporting Service та відповідного SDK Visual Studio.

Також ще існують модулі валідації даних після запису екземплярів сутностей. Вони не обмежені функціями валідаторів, а навпаки можуть містити пост функції створення чи оновлення існуючих у системі записів.

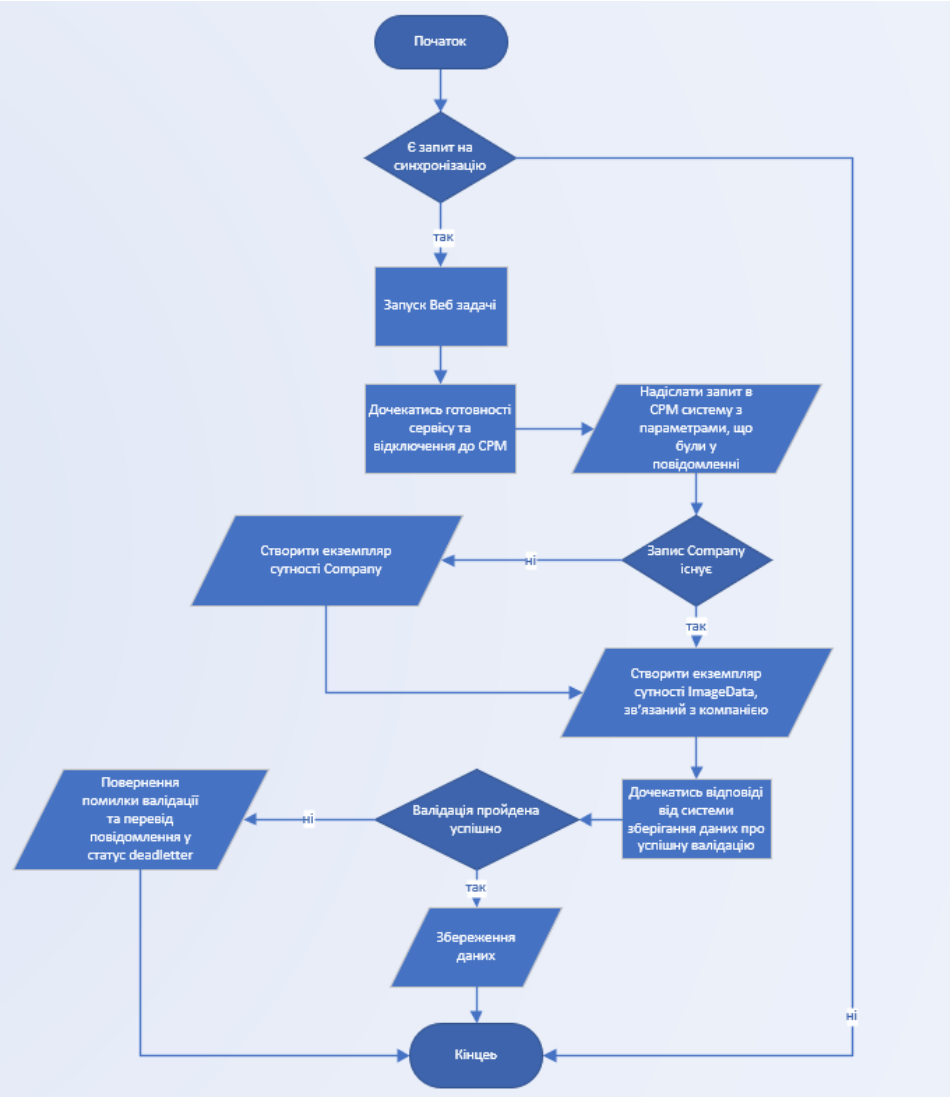

Рисунок 7.3 – Блок-схема модуля обробки даних

Висновки до розділу

Через модульність кодування всієї системи поділяється на кілька етапів. Перший крок у розробці – це створення інфраструктури, необхідної для забезпечення працездатності функціонального модуля в процесі розробки.

Сам функціональний модуль кодується в порядку, що відповідає логічному потоку даних в системі.

В даному розділі розглянуто основні, найважливіші модулі обробки даних, серед таких модуль обробки даних, модуль формування звітів, веб-задача. Ці модулі несуть на собі задачі інтеграції, обробки даних, що приходять звідти, а також запис їх до сховища даних, представленого SQL базою.

Також детально розписано розгортання інстансу CRM системи, перенос рішень для неї та усіх необхідних логічних модулів. Крім того, в розділі наявна інструкція по налаштуванню середовища з чергою, в якій зберігаються повідомлення для обробки.

Останнім кроком у розробці системної логіки є публікація та налаштування всіх функціональних модулів у хмарному середовищі. Маючи на увазі хмарну технологію, публікуйте її перед завершенням проекту, щоб виявити та вирішити помилки інтеграції на ранньому етапі.

8 РОЗРОБКА СТАРТАП-ПРОЕКТУ

Стартап – це унікальна ідея, що спрямована на вирішення проблем, що наявні у певній сфері, а також на отримання прибутку у ресурсах, фінансах чи інших величинах. Найчастіше стартап це щось нове, інноваційне, певне рішення, якого раніше не було. Проте так само найчастіше важко запропонувати щось абсолютно нове, тому частіше стартап – це унікальна ідея або спосіб використання, певний підхід у вирішенні проблематики.

8.1 Опис ідеї проекту

Проект: використання налаштованої системи для зберігання та обробки даних з алгоритмами аналітичного аналізу. Особливостями буде інтеграція з системою розпізнавання образів, наявність звітів по роботі з зовнішньою системою, по роботі персоналу у системі, компоненти бізнес-логіки обробки даних.

Таблиця 8.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки використання	Вигоди для користувача
	1. Підвищення ефективності ведення бізнесу	Ефективне зберігання даних дозволить ефективно проводити політику компанію щодо надання послуг клієнтам, швидкодія системи та можливості для модифікація зекономлять час на розробку додаткових модулів та модифікацію існуючих.
	2. Аналіз ефективності використання ресурсів	Алгоритми підсумкового аналізу дозволять ефективніше оцінювати роботу персоналу, а також інших ресурсів, як час або майнові ресурси.

Основними техніко-економічними характеристиками ідеї є:

- зберігання та обробка даних;
- інтеграція з системою розпізнавання образів;
- звіти та аналітика;
- широка функціональність по налаштуванню процесу праці;
- можливість налаштування системи під користувача;
- доступ до системи онлайн;
- перегляд подій та дій користувачів;

Є різноманітні системи серед конкурентів. Серед найближчих по функціональності конкурентів можна зазначити системи salesforce, amoCrm, bpm`online. Для визначення сильних, нейтральних та слабких характеристик системи, було здійснено збір та аналіз інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку. Таблиця 8.2 містить порівняння даної системи із конкурентами, в ній стовпчик W – слабка сторона, N - нейтральна сторона, S - сильна сторона.

Таблиця 8.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту.

№	Техніко-економічні характеристики ідеї	Потенційні товари/концепції конкурентів				W	N	S
		Мій проект	salesforce	amoCrm	bpm`online			
1.	Зберігання та обробка даних	Має	Має	Має	Має	-	+	-
2.	Інтеграція з системою розпізнавання образів	Має	Немає	Немає	Немає	-	-	+
3.	Наявність звітів та аналітики	Має	Має	Має	Немає	-	-	+
4.	Можливість налаштування бізнес процесів під користувача	Має	Немає	Має	Має	-	-	+

Продовження таблиці 8.2

5.	Доступ до системи онлайн	Має	Має	Має	Має	-	+	-
6.	Широка функціональність по налаштуванню процесу праці	Має	Має	Немає	Має	-	-	+
7.	Перегляд подій та дій користувачів	Має	Немає	Немає	Немає	-	-	+

10.2 Технологічний аудит ідеї проекту

Для вдалої реалізації проекту, необхідно провести огляд існуючих технологій та аналіз на степінь доступності, готовності до використання. По результатам цього, був обраний необхідний набір технологій, встановлена їх доступність. Таким чином була встановлена можливість технологічної здійсненності ідеї проекту, і як підсумок оформлена у таблиці 8.3

Таблиця 8.3 – Технологічна здійсненність ідеї проекту.

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1.	Зберігання та обробка даних	Microsoft SQL	Наявна	Доступна
2.	Інтеграція з системою розпізнавання образів	Система з використанням Веб-фреймворку Asp.net core	Наявна	Доступна

Продовження таблиці 8.3

3.	Наявність звітів та аналітики	SQL Reporting Service	Наявна	Доступна
4.	Доступ до системи онлайн	Wifi, доступ до мережы Internet	Наявна	Доступна
5.	Модифікація вбудованих алгоритмів та процесів	Мова програмування C#, fetchXml, JS	Наявна	Доступна
6.	Серверна частина	Asp.net core	Наявна	Доступна
7.	Безпечний зв’язок	Протокол HTTPS	Наявна	Доступна
8.	Збереження даних сервері	Microsoft SQL Server	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: є можливою.				

Згідно з проведеним дослідженням, перераховані технології легко доступні до використання та можливостей побудови систем, методи реалізації є доступними, деякі з технологій є платними.

8.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можливо використати під час ринкового впровадження проекту, а також та загроз, що можуть перешкоджати процесу реалізації проекту. Це дозволяє планувати напрямки розвитку проекту із урахуванням стану середовища впровадження, а також потреб потенційних клієнтів та пропозицій конкурентів. Для запуску стартап-проекту необхідно визначити стан середовища впровадження, яким є ринок, визначити загрози, спланувати напрями розвитку, потреби потенційних клієнтів та пропозицій конкурентів. Попередня характеристика потенційного ринку зображена на таблиці 8.4.

Таблиця 8.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	1 (розробники)
2.	Загальний обсяг продаж, грн/ум. од.	Понад 2000
3.	Динаміка ринку (якісна ціна)	Зростає
4.	Наявність обмежень для входу	Конкуренція
5.	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6.	Середня норма рентабельності в галузі (або по ринку), %	170%

Оскільки демографія зростає, зростають доходи населення, то ринок збільшується. І за результатами проведеного дослідження, для входження ринок є привабливим, а норма рентабельності є задовільною.

Надалі, необхідно визначити потенційних клієнтів, їх характеристики та сформувати перелік вимог до кожного виду товару. Характеристика описана в таблиці 8.5.

Таблиця 8.5 – Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Потреба в підвищенні ефективності	Представники середнього та великого бізнесу	Поведінка клієнта формується від об’ємів надання	- висока якість - легкість у використанні та швидке

роботи персоналу та роботи з клієнтами, отримання аналітики в процесі роботи		послуг, а також від кількості партнерів клієнта	навчання - швидкодія системи - надійність роботи - зручність використання

Внаслідок проведення аналізу стосовно потенційних груп клієнтів, було встановлено, що основна кількість клієнтів це представники середнього та великого бізнесу. Тому зосередитись важливо на основних вимогах даної групи споживачів, в першу чергу швидкодія системи та зручність використання, а також ефективна аналітика праці. Після визначення вимог клієнтів необхідно провести огляд ринкового середовища, розглянути фактори, які сприяють впровадженню проекту на ринку, та фактори, що йому перешкоджають (табл. 8.6, табл. 8.7).

Таблиця 8.6 – Фактори загроз стартап-проекту

№	Фактор	Зміст загрози	Можлива реакція компанії
1.	Залежність від постачальника послуг	Є цінова залежність від вартості підписок на використання технологій, що використовуються. Дані підписки надаються зовнішньою компанією	Пошук інших постачальників, створення універсальної системи

Продовження таблиці 8.6

2.	Виявлення нових технологій в наслідку наукового прориву	Поява кращих технологій, компонентів	Використання нових технологій чи компонентів у нових моделях/продуктах компанії
3.	Поява нової системи-конкурента на ринку	Поява конкурентів	Докладний аналіз продукту конкурента, адаптація проекту
4.	Складність налаштування	Програмний продукт складно зробити універсальним для будь-якого підприємства чи компанії	Реалізація базового спільного функціоналу та пропозиція налаштування системи унікально під користувача
5.	Мала кількість продажів	Прогнозовані продажі не виправдались	Пошук нових можливостей покращення продукту, збирання та дослідження відгуків користувачів
6.	Світова криза	Частково може зменшити кількість продаж через високу вартість системи, проте також може бути каталізатором, оскільки в умовах кризи клієнти можуть бути у пошуках ефективності	Пошуки шлях здешевлення, формування базового рішення з більшим функціоналом, щоб зекономити на налаштуванні системи під користувача

Продовження таблиці 8.6

7.	Нерозуміння користі	Деяким клієнтам, не зрозуміла важливість ефективної аналітики та даної системи загалом, оскільки на їх думку вартість продукту не виправдовує отриманого результату	Проведення опитування вже існуючих клієнтів для отримання зворотного зв'язку по збільшенню ефективності після впровадження

Таблиця 8.7 – Фактори можливостей стартап-проекту

№	Фактор	Зміст можливості	Можлива реакція компанії
1.	Зацікавленість в продукті іноземних клієнтів	Розширення ринку, нові клієнти	Дороблення системи для задоволення вимог нових клієнтів
2.	Зацікавленість у продукті спонсорів або інвесторів	Збільшення капіталу проекту	Масштабування продукту, збільшення маркетингу
3.	Збільшення кількості представників середнього бізнесу	Розширення кількості версій, ринку, нові клієнти	Запровадження модифікацій системи в залежності від видів тварин

Для проекту системи зберігання та обробки інформації було проаналізовано та встановлено основні фактори загроз, а також можливостей. Існують загрози різних напрямків та варіацій. Більшість загроз це зовнішні фактори, до яких необхідно пристосовуватись по мірі необхідності та по мірі появи нових загроз. Проте ймовірність появи суттєвих загроз для проекту є незначною, а реальних можливостей значно більше ніж потенційних загроз. Враховуючи це, проект має потенціал для розвитку на ринку, а

також має потенціал до розвитку. Також необхідно визначити риси конкуренції на ринку (табл. 8.8).

Таблиця 8.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив діяльності підприємства
Тип конкуренції: чиста	Конкурентами є компанії із приблизно однаковою долею ринку	Налаштування системи швидше, якісніше, із використанням більш зручного функціоналу, нижчою ціною, з аналітичною складовою
За рівнем конкурентної боротьби: світовий	Ринок охоплює всю планету	Забезпечення відповідності продукту світовим стандартам
За галузевою ознакою: внутрішньо-галузева	Конкуренти знаходяться в одній галузі	Аналіз факторів, які впливають на успіх у даній галузі
За характером конкурентних переваг: цінова, не цінова	Система якісніша, а також дешевша за конкурентів	Вкладання зусиль, ресурсів у підтримання вищої якості продукту
За інтенсивністю: не марочна	Система тільки виходить на ринок	Забезпечити успішний старт продукту на ринку

Завдяки ступеневому аналізу конкуренції на ринку, можна зробити висновок, що конкурентні товари існують на ринку на різних позиціях та їх кількість чимала, проте

сам ринок є перспективним, оскільки товари та ідеї конкурентів мають недоліки, а також кількість конкурентів значно менша кількості споживачів, яких вони можуть обслуговувати. Більш детальний аналіз умов конкуренції в галузі - далі (табл. 8.9).

Таблиця 8.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потецнійні конкуренти	Постачальники	Клієнти	Товари-замінники
	Системи salesforce amoCrm bpm`online	Конкурентом може стати будь-який продукт що вирішуватиме задачу зберігання та обробки інформації	Деякі постачальники мають підконтрольною велику частину ринку	Представники середнього та великого бізнесу	Немає
Висновки	Інтенсивність конкурентної боротьби доволі висока	Потенційні конкуренти можуть виникнути згодом	Існує сильна залежність від постачальників, проте вони не диктують умови роботи ринку.	Головною вимогою клієнта є ефективність роботи системи	Немає ризику

Аналіз за М. Портером показав, що наявні декілька прямих конкурентів та є можливість виникнення потенційних конкурентів. Але ситуація загалом сприятлива, оскільки немає ризику товарів-замінників. Головною вимогою клієнтів є ефективність роботи. Хоча й існує залежність від постачальників але все ж проект може мати успіх.

Для досягнення успіху варто визначити сильні та слабші сторони проекту. Обґрунтування факторів конкурентоспроможності описано в таблиці 8.10

Таблиця 8.10 – Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування факторів
1.	Віддалене керування	Продукт дає можливість керування з будь-якої точки доступу використовуючи мережу Інтернету
2.	Зручність використання	Продукт економить час клієнта, автоматизуючи процес управління бізнес процесами
3.	Комплексний підхід	Поєднання усього функціоналу в одному місці покращує взаємодію користувача з системою
4.	Можливість розширення продукту	Продукт передбачає можливість розширення, модифікації функціоналу під користувача, а також наявні інструменти налаштування процесів фвласноруч користувачем системи
5.	Наявність алгоритмів підсумкового аналізу	Використання звітів дозволить оцінювати ефективність роботи працівників, окремих об’єктів бізнесу, а також підвищити ефективність надання послуг споживачам або партнерам компанії
6.	Можливість налаштування ролей безпеки	Дана можливість присутня у основі для нашої системи – dynamics 365
7.	Інтеграція з автоматизованою системою розпізнавання образів	Дана інтеграція дозволить, за допомогою моделі даних, підключати потужні алгоритми обробки даних про об’єкти та ефективно використовувати їх для обробки та аналізу

Щоб гарантовано мати успіх у сучасній ринковій ситуації, для проекту були обрані фактори, на яких буде заснована його перевага над конкурентами. Найважливішими серед факторів конкурентоспроможності є алгоритми підсумкового аналізу, а також інтеграція із автоматизованою системою розпізнавання образів. Порівняння факторів конкурентоспроможності дає можливість визначити сильні та слабкі сторони стартап-проекту. bpm`online – BO, salesforce – SF, amoCRM - AC (табл. 10.11).

Таблиця 8.11 – Порівняльний аналіз сильних і слабких сторін стартап-проекту

Фактор конкурентос-проможності	Бали 1-20	Рейтинг товарів-конкурентів порівняно з проектом						
		-3	-2	-1	0	+1	+2	+3
Зручність використання	10		SF	BO				AC
Можливість розширення проекту	18		SF	amo CR M		BO		
Наявність аналітики	10		AC		BO	SF		
Інтеграція з системою розпізнавання образів	20	SF, AC, BO						

По результатам порівняльного аналізу можна зазначити, що згідно факторів конкурентоспроможності, значні переваги над наявними конкурентами має саме стартап-проект, завдяки чому віно має всі можливості виходу на ринок.

Для повноти аналізу, необхідно складання SWOT-аналізу на основі ринкових можливостей та загроз, а також сильних і слабких сторін (табл 10.12).

Таблиця 8.12 – SWOT-аналіз стартап-проекту

Сильні сторони: - актуальність проекту; - функції, яких немає у конкурентів; - можливість підвищення ефективності при використанні - зручність використання системи.	Слабкі сторони: - багато конкурентів на ринку - складність масштабного впровадження; - наявна певна залежність від постачальників послуг.
Можливості: - поява інвесторів; - розширення ринку шляхом зацікавленості продуктом; - послаблення позицій конкурентів; - впровадження нових технологій першими.	Загрози: - проблеми з постачальниками; - поява нової системи-конкурента на ринку; - мала кількість продажів; - висока собівартість;

Як наслідок проведення SWOT-аналізу, були визначені варіанти сильні та слабкі сторони проекту, а також визначені можливі наслідки, як можливості, так и загрози (табл. 10.13). SWOT-аналіз дав змогу оцінити наш продукт з різних сторін, а також зробити порівняльний аналіз характеристик продукту та ринкового середовища.

Таблиця 8.13 – Альтернативи ринкового впровадження

Альтернатива (орієнтов-ний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
Загарбник	Середня	Більше 2 років
Наступник	Суттєва	1-2 роки
Виклик лідеру	Низька	Більше 3 років

Для нашого проекту були визначені основні три альтернативи ринкової поведінки – загарбник, наступник, виклик лідеру. При використанні альтернативи Загарбник існує середня ймовірність отримання ресурсів, оскільки даний пункт залежить від найбільшої кількості факторів, ніж усі запропоновані. При використанні альтернативи наступник існує високий шанс отримання ресурсів завдяки захопленню конкретної сфери. Остання альтернатива виклик лідеру скоріш за все принесе ресурси, проте терміни їх отримання збільшені, оскільки лідери вже мають клієнтську базу, найчастіше з позитивним досвідом співпраці, а також певний імідж.

8.4 Розроблення ринкової стратегії проекту

Для реалізації проекту на ринку, необхідно розробити положеня, що коротко описують потенційних споживачів, цільовий ринок, спосіб позиціонування товару і величини обсягів продажу, яких планується досягнути за перші кілька років реалізації товару. Для цього зробимо перший крок визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 10.14).

Таблиця 8.14 – Вибір цільових груп потенційних споживачів

Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтований попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу в сегмент
Великий бізнес	Висока	Високий	Висока	Середня
Середній бізнес	Середня	Середній	Висока	Середня
Які цільові групи обрано: середній та заможний класи населення				

За результатами аналізу потенційних груп споживачів (сегментів) можна зазначити,

що групами, що визначені цільовими, є середній та великий бізнес, що складають приблизно половину із усіх бізнес клієнтів, залежно від країни оцінки, тому доцільним є обрання стандартизовану програму масового маркетингу.

Наступний крок розробки ринкової стратегії полягає в визначенн базової стратегії розвитку (табл. 8.15).

Таблиця 8.15 – Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Виклик лідеру	Концентрація на потребах великого сегменту ринку	Надання клієнтам сучасного програмного забезпечення за зниженою ціною	Стратегія диференціації

З відомих стратегій розвитку для першого періоду стартапу була обрана стратегія диференціації, як така, що найбільш підходить для умов ринку з урахуванням характеристик програмного продукту. Виокремлення найвагоміших факторів та характеристик нашого продукту серед конкурентів і є метою. Наступним пунктом в розробці ринкової стратегії полягає у виборі стратегії конкурентної поведінки (табл. 8.16).

Таблиця 8.16 – Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Проект є наступником вже існуючих проектів	Проект буде заохочувати нових споживачів до співпраці	Необхідно реалізувати такі характеристики конкурентів: - налаштування бізнес-ролей; - велика кількість вбудованих бізнес процесів; - зручний інтерфейс користувача	Стратегія виклику лідера.

Отже, як базову стратегію конкурентної поведінки обираємо стратегію виклиу лідеру. Серед представлених, дана стратегія найкраще відповідає меті проекту Суть її полягає у витісненні лідерів з даної сфери та займання їх позицій на ринку. Наступним кроком є визначення стратегії позиціонування (табл. 10.17).

Таблиця 8.17 – Визначення стратегії позиціонування

Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
Висока ефективність, наявність алгоритмів підсумкового аналізу, зручний інтерфейс, реалізація необхідних для ведення бізнесу процесів	Стратегія диференціації	Стратегія виклику лідера	Ефективність, аналітика, підвищення доходів

Таким чином, позиціонування продукту було визначено на основі SWOT-аналізу, вибору цільових груп потенційних споживачів, визначення стратегії захоплення ринку та конкурентоспроможних переваг продукту.

8.5 Розроблення маркетингової програми стартап-проекту

Для того щоб швидко та успішно реалізувати проект, наступним кроком є розробка маркетингової програми, яка є системою взаємозалежних заходів, що визначаються поведінкою виробника протягом комісійного періоду. Формування маркетингової програми базується на даних про відповідні вимоги клієнтів, визначених потребами

нинішніх та потребами майбутніх потенційних клієнтів, використовуючи маркетингові стратегічні стратегії та тактику. Основним кроком є формування маркетингової концепції для продуктів, які отримують клієнти.

Необхідно здійснити визначення ключових переваг концепції потенційного товару (табл. 8.18).

Таблиця 8.18 – Визначення ключових переваг концепції потенційного товару

Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
Потреба в економії часу	Зручний у використані сучасний програмний продукт, що має переваги над конкурентами	Ключовими перевагами є: Ефективність налаштування бізнес процесів, зручність
Потреба в отриманні аналітики	Можна буде формувати звіту та ефективно аналізувати етапи роботи, а також оцінювати ефективність роботи персоналу	Зручність використання системи звітності

Ключовими перевагами даного проекту є ефективність, наявність алгоритмів підсумкового аналізу, зручний інтерфейс, реалізація необхідних для ведення бізнесу процесів. Подальшим кроком є розробка трирівневої маркетингової моделі товару, для уточнення ідеї продукту або послуги, особливості процесу надання, його фізичні складові (табл. 8.19).

Таблиця 8.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td></td><td colspan="3">Опис базової потреби споживача, яку задовольняє товар (згідно концепції), її основної функціональної вигоди</td></tr>
<tr><td></td><td colspan="3">Продаж системи обробки та зберігання даних клієнтам, що дозволить більш ефективно розпоряджатись ресурсами</td></tr>
<tr><td rowspan="5"></td><td>Властивості/ характеристики</td><td>М/Нм</td><td>Вр/Тх/Тл/Е/Ор</td></tr>
<tr><td>1. Економічні: вартість планується зменшити за рахунок реалізації меншої кількості бізнес процесів
2. Технологічні: використання сучасних технологій
3. Ергономічність: зручний користувацький інтерфейс, зрозумілий функціонал продукту
4. Естетичні: інтерфейс користувача приємний
5. Безпеки: відповідність нормам використання електронних пристроїв
6. Надійності: Система має надійно працювати без збоїв</td><td>–/+</td><td>+/+/+/+/+</td></tr>
<tr><td colspan="3">Якість: система має працювати однаково, без некоректного відпрацювання</td></tr>
<tr><td colspan="3">Документи виконані з логотипом підприємства</td></tr>
<tr><td colspan="3">Марка: boxCrm</td></tr>
</table>

Продовження таблиці 8.19

III. Товар із підкріпленням	До продажу: представлення клієнтам продукту
	Після продажу: допомога в налаштуванні, ремонт, підтримка продукту
За рахунок чого потенційний товар буде захищено від копіювання: використання електронних підписів у збірках, а також використання ключів для публікації	

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар, яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 10.20).

Таблиця 8.20 – Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
-	5000-50000 ум. од.	100000 ум. од.	3000 ум. од.

Оскільки цільових груп споживачів – дві, але одна значно більша за іншу, то при встановленні цін, варто керуватися можливостями більшої групи. Орієнтовна ціна сформована згідно можливостей середнього бізнесу, відповідно для великого бізнесу дані ціни будуть прийнятними, а отже товар буде доступний для усіх типів користувачів. Наступним кроком є формування системи збуту (10.21).

Таблиця 8.21 – Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
Покупка після презентації товару	Доставка, надання консультацій при продажі та використанні	Глибока	Власні сили, платформи для продажу рішень

Для оптимальності системи збуту було вирішено проводити продаж власними силами, використовувати інтернет-майданчики для розповсюдження товару серед яких є представлені великі магазини продажу, такі як Microsoft Store та інші. Останнім кроком маркетингової програми є розроблення маркетингових комунікацій (табл. 10.22). Маркетингові комунікації надзвичайно важливий компонент розвитку стартап проекту, оскільки від нього залежить в більшості залежить успіх проекту на початкових стадіях. На наступних стадіях маркетингові комунікації вийдуть на інший рівень і від пошуку нових споживачів вони вийдуть на рівень покращення стосунків з уже існуючими споживачами та отримання нових клієнтів за рахунок сарафанного радіо.

Таблиця 8.22 – Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення

Продовження таблиці 8.22

Серед клієнтів багато менеджерів шукають рішення для своїх проблем на спеціалізованих інтернет-майданчиках	Інтернет, Сарафанне радіо	Сучасна система дбає, що сприяє ефективному веденню бізнесу	Донести користувачам інформацію про якісний і сучасний продукт з перевагами	Оригінально сповістити споживачів про новий продукт

Основним каналом комунікації для користувачів, є Інтернет, і використання цього каналу є необхідним. Маркетингова кампанія в мережі дозволяє охопити більш широку аудиторію і зацікавити більше клієнтів. Ключовими позиціями продукту є його ефективність та певний унікальний функціонал.

10.6 Висновки до розділу

Даний стартап проект є налаштованою під користувача системою зберігання та обробки даних з використанням алгоритмів підсумкового аналізу. Дана система при використанні дає можливість підняти ефективність роботи персоналу, а також зробити процес праці простішим. Можливість ведення аналітики за допомогою системи звітів дозволить об’єктивно оцінити ефективність роботи системи та працівників із системою. В системі автоматизовано основні бізнес-процеси однакові для більшості підприємств, тому їхнє використання може бути зручним для більшості клієнтів. Також існує можливість налаштування додаткових бізнес процесів для користувача за необхідності. У базовій системі наявний широкий вибір компонентів для розробки та налаштування.

На ринку існую декілька близьких за функціональністю конкурентів. Чітко були визначені сильні, а також нейтральні та слабкі сторони системи, а також систем конкурентів. Дане порівняння виявило, що розроблюваний проект має перспективу у

розвитку. Технічний аудит проекту допоміг виявити, що технології, необхідні для реалізації проекту доступні на ринку для впровадження.

Згідно дослідження потенційного ринку було проведено дослідження ринку. Як висновок, можна зазначити, що ринок є привабливим для входу проекту. Норма рентабельності достатньо висока.

Проведений аналіз конкурентів виявив, що представники середнього та великого бізнесу є цільовими категоріями споживачів для розроблюваного проекту. Система цілком задовольняє основні потреби по підвищенню ефективності та отримання ефективної аналітики.

Також були розглянуті загрози різноманітного характеру, що пов’язані з різними факторами. Два найголовніші з них це конкуренти та постачальники, а також розвиток нових технологій та оновлення старих. Аналіз виявив, що загрози не є критичними та шанси їх виникнення також не високі. В той час як можливості навпаки сприяють впровадженню проекту, вірогідність кінцевого успіху доволі висока. Також були визначені фактори, на яких основана перевага над конкурентами. Найголовнішими такими факторами є ефективне зберігання та обробка даних, також система звітності з використанням алгоритмів підсумкового аналізу.

Проект має можливості та перспективи для масштабування, оскільки клієнтів велика кількість і їхні запити специфічні відповідно до сфери ведення діяльності. Тому існує багато простору для надання додаткових послуг, а саме налаштування бізнес процесів індивідуально під користувача. Але для першого етапу була обрана стратегія диференціації, яка була визначена, як найбільш ефективна в даних ринкових умовах. Із збільшенням клієнтської бази та продаж планується розробка подальшої стратегії розвитку.

Маркетингова програма продукту дозволить своєчасно та успішно реалізувати проект. У маркетинговій програмі продукту було використано ключові переваги концепції даного проекту, а саме можливість системи при використанні надати можливість підняти ефективність роботи персоналу, а також зробити процес праці

простішим та можливість ведення аналітики за допомогою системи звітів дозволить об’єктивно оцінити ефективність роботи системи та працівників із системою. Позиціонування товару буде засноване на його ключових відмінних характеристиках

Встановлені цінові межі були обґрунтованими для цільових груп клієнтів та можливі для досягнення. Система збуту визначена Інтернет продажами та інтернет продажами на спеціалізованих інтернет-майданчиках.

Оптимальними каналами для комунікації зі споживачами обрано Інтернет та відповідні спеціалізовані ресурси.

ВИСНОВКИ

У дипломному проекті була розроблена автоматизована система обробки та зберігання даних з використанням алгоритмів підсумкового аналізу.

Система є модульною, ефективною, а також функціонально насиченою, завдяки чому має всі можливості для виходу на ринок в умовах навіть сильного рівня конкуренції.

В ході виконання даного проекту був проведений поглиблений аналіз існуючих рішень, предметної області, проаналізовано існуючі проблеми, а також визначили основні напрями руху проекту, в якому напрямку потрібно рухатись та якої мети досягти у процесі. Найбільш суттєвим недоліком сучасних рішень конкурентів є відсутність ефективної аналітики у системах зберігання даних. Як корисне доповнення було реалізовано інтеграцію з системою розпізнавання образів для підвищення ефективності ведення бізнес процесів.

Всі ці результати стали можливими завдяки дослідженню та визначенню функціональних особливостей системи, а також не функціональних, а також сформовані основні вимоги до системи. Розроблені та визначені основні процеси для основних бізнес ролей, а саме імпортера, оператора та адміністратора.

Була розроблена багаторівнева структура системи з використанням багатого стеку технологій та мов програмування. Обрані платформи розробки дозволили створити гнучку систему, що складається з окремих модулів взаємодії на різних програмних рівнях в межах системи та поза її межами. Основними двома платформами було обрано .Net Framework та .Net Core. Дані платформи схожі та використовують однакову мову програмування та підходи у розробці, проте мають свої функціональні особливості, згідно яких їх можна використовувати ефективно в рамках різних підсистем даного проекту.

На основі розробленої системи та її характеристик було проведено дослідження потенціального виходу на ринок у вигляді стартап-проекту. Проведено глибокий аналіз

ідеї, основних характеристик програмного продукту, а також сильних сторін системи, що роблять її конкурентноспроможною. Результатом проведеного дослідження є висновок, що дана система є конркурентоспроможною, здатною до виходу на світовий ринок та такою, що може приносити прибуток.

Подальшою сферою роботи або напрямком роботи системи є модульність. Використання мікросервісів, що зможуть зробити систему ще більш гнучкою, використання нових технологій у формуванні алгоритмів підсумкового аналізу. Як наслідок рівень ефективності використання системи може суттєві підвищитись.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Habr.com [Електронний ресурс] : Режим доступу: https://habr.com/ru/company/oleg-bunin/blog/348172/

2. .NET Core [Електронний ресурс] : Режим доступу: https://dotnet.microsoft.com

3. Round 18 results – TechEmpower Framework Benchmark [Електронний ресурс] : Режим доступу: https://www.techempower.com/benchmarks/#section=data-r18&hw=ph&test=plaintext

4. Azure SDK for .NET [Електронний ресурс] : Режим доступу: https://docs.microsoft.com/en-us/dotnet/azure/dotnet-tools?view=azuredotnet&tabs=windows

5. Service Bus Explorer [Електронний ресурс] : Режим доступу: https://github.com/paolosalvatori/ServiceBusExplorer

6. SSRS [Електронний ресурс] : Режим доступу: https://docs.microsoft.com/en-us/sql/reporting-services/tools/tutorial-how-to-locate-and-start-reporting-services-tools-ssrs?view=sql-server-ver15

7. Data migration [Електронний ресурс] : Режим доступу: https://docs.microsoft.com/ru-ru/dynamics365/admin/manage-configuration-data

8. Solution management [Електронний ресурс] : Режим доступу: http://mmcrm.ru/?p=884

9. System architecture [Електронний ресурс] : Режим доступу: https://studopedia.su/10_69171_bagatorivnevi-sistemi.html

10. Архітектура програмного забезпечення [Електронний ресурс] : Режим доступу: https://uk.wikipedia.org/wiki/

11. Багаторівнева архітектура програмного додатку [Електронний ресурс] : Режим доступу: https://metanit.com/sharp/mvc5/23.5.php

12. HTML [Електронний ресурс] : Режим доступу: https://ru.wikipedia.org/wiki/HTML

13. JS [Електронний ресурс] : Режим доступу: https://ru.wikipedia.org/wiki/JavaScript

14.Об'єкти першого класу [Електронний ресурс] : Режим доступу: https://ru.wikipedia.org/wiki/

15. Мова налаштування стилів [Електронний ресурс] : Режим доступу: https://ru.wikipedia.org/wiki/CSS

ДОДАТОК А

ДОДАТОК Б

ДОДАТОК В

ДОДАТОК Г

ДОДАТОК Д

ДОДАТОК Е

ДОДАТОК Ж

ДОДАТОК И